21世纪高等学校规划教材 | 软件工程





Web工程
——理论与实践

霍秋艳 徐学洲 陈静玉 何昊 等编著

清华大学出版社

21世纪高等学校规划教材·软件工程

Web工程——理论与实践

霍秋艳　徐学洲　陈静玉　何　昊　等编著

清华大学出版社
北　京

内容简介

本书系统地介绍了综合而系统化地开发高质量的Web应用的原理、方法、技术和工具。内容涵盖Web应用的特征、Web工程开发过程和Web项目管理、Web项目需求工程、Web应用建模、Web应用架构、Web应用设计、Web应用开发和部署、测试、运行和维护、Web应用的性能、可用性、安全性，以及Web工程未来的发展趋势，案例贯穿全书。

本书可以作为计算机软件专业或计算机相关专业的本科生与研究生的教材，也适合作为相关专业人员的参考书。

图书在版编目(CIP)数据

Web工程：理论与实践/霍秋艳等编著.—北京：清华大学出版社，2011.8
(21世纪高等学校规划教材·软件工程)
ISBN 978-7-302-25697-7

Ⅰ.①W…　Ⅱ.①霍…　Ⅲ.①网页制作工具—程序设计—高等学校—教材
Ⅳ.①TP393.092

中国版本图书馆CIP数据核字(2011)第103155号

责任编辑：闫红梅　李玮琪
责任校对：焦丽丽
责任印制：

出版发行：清华大学出版社　　**地　　址：**北京清华大学学研大厦A座
http://www.tup.com.cn　　**邮　　编：**100084
社 总 机：010-62770175　　**邮　　购：**010-62786544
投稿与读者服务：010-62795954，jsjjc@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 刷 者：
装 订 者：
经　　销：全国新华书店
开　　本：185×260　　**印　　张：**19.75　　**字　　数：**481千字
版　　次：2011年　月第1版　　**印　　次：**2011年　月第1次印刷
印　　数：1～　000
定　　价：　.00元

产品编号：038254-01

编审委员会成员

（按地区排序）

出版说明

随着我国改革开放的进一步深化，高等教育也得到了快速发展，各地高校紧密结合地方经济建设发展需要，科学运用市场调节机制，加大了使用信息科学等现代科学技术提升、改造传统学科专业的投入力度，通过教育改革合理调整和配置了教育资源，优化了传统学科专业，积极为地方经济建设输送人才，为我国经济社会的快速、健康和可持续发展以及高等教育自身的改革发展做出了巨大贡献。但是，高等教育质量还需要进一步提高以适应经济社会发展的需要，不少高校的专业设置和结构不尽合理，教师队伍整体素质亟待提高，人才培养模式、教学内容和方法需要进一步转变，学生的实践能力和创新精神亟待加强。

教育部一直十分重视高等教育质量工作。2007 年 1 月，教育部下发了《关于实施高等学校本科教学质量与教学改革工程的意见》，计划实施"高等学校本科教学质量与教学改革工程"（简称"质量工程"），通过专业结构调整、课程教材建设、实践教学改革、教学团队建设等多项内容，进一步深化高等学校教学改革，提高人才培养的能力和水平，更好地满足经济社会发展对高素质人才的需要。在贯彻和落实教育部"质量工程"的过程中，各地高校发挥师资力量强、办学经验丰富、教学资源充裕等优势，对其特色专业及特色课程（群）加以规划、整理和总结，更新教学内容、改革课程体系，建设了一大批内容新、体系新、方法新、手段新的特色课程。在此基础上，经教育部相关教学指导委员会专家的指导和建议，清华大学出版社在多个领域精选各高校的特色课程，分别规划出版系列教材，以配合"质量工程"的实施，满足各高校教学质量和教学改革的需要。

为了深入贯彻落实教育部《关于加强高等学校本科教学工作，提高教学质量的若干意见》精神，紧密配合教育部已经启动的"高等学校教学质量与教学改革工程精品课程建设工作"，在有关专家、教授的倡议和有关部门的大力支持下，我们组织并成立了"清华大学出版社教材编审委员会"（以下简称"编委会"），旨在配合教育部制定精品课程教材的出版规划，讨论并实施精品课程教材的编写与出版工作。"编委会"成员皆来自全国各类高等学校教学与科研第一线的骨干教师，其中许多教师为各校相关院、系主管教学的院长或系主任。

按照教育部的要求，"编委会"一致认为，精品课程的建设工作从开始就要坚持高标准、严要求，处于一个比较高的起点上。精品课程教材应该能够反映各高校教学改革与课程建设的需要，要有特色风格、有创新性（新体系、新内容、新手段、新思路，教材的内容体系有较高的科学创新、技术创新和理念创新的含量）、先进性（对原有的学科体系有实质性的改革和发展，顺应并符合 21 世纪教学发展的规律，代表并引领课程发展的趋势和方向）、示范性（教材所体现的课程体系具有较广泛的辐射性和示范性）和一定的前瞻性。教材由个人申报或各校推荐（通过所在高校的"编委会"成员推荐），经"编委会"认真评审，最后由清华大学出版

社审定出版。

目前，针对计算机类和电子信息类相关专业成立了两个“编委会”，即“清华大学出版社计算机教材编审委员会”和“清华大学出版社电子信息教材编审委员会”。推出的特色精品教材包括：

(1) 21世纪高等学校规划教材·计算机应用——高等学校各类专业，特别是非计算机专业的计算机应用类教材。

(2) 21世纪高等学校规划教材·计算机科学与技术——高等学校计算机相关专业的教材。

(3) 21世纪高等学校规划教材·电子信息——高等学校电子信息相关专业的教材。

(4) 21世纪高等学校规划教材·软件工程——高等学校软件工程相关专业的教材。

(5) 21世纪高等学校规划教材·信息管理与信息系统。

(6) 21世纪高等学校规划教材·财经管理与应用。

(7) 21世纪高等学校规划教材·电子商务。

(8) 21世纪高等学校规划教材·物联网。

清华大学出版社经过三十多年的努力，在教材尤其是计算机和电子信息类专业教材出版方面树立了权威品牌，为我国的高等教育事业做出了重要贡献。清华版教材形成了技术准确、内容严谨的独特风格，这种风格将延续并反映在特色精品教材的建设中。

清华大学出版社教材编审委员会
联系人：魏江江
E-mail：weijj@tup.tsinghua.edu.cn



Web是一种快速增长的新型通信媒体，自从1989年问世以来，Web技术发展迅速，其范围、功能和应用飞速增长，已成为新千年IT领域发展的柱石之一。工业、商业、学术界、政府和个人大量使用着Web，Web已经以各种形式融入并影响着大多数人的日常生活和工作。Web应用（基于Web的系统）开发已经不再只是关注开发技术，越来越需要系统化地开发复杂Web应用的原理、过程、方法以及技术，需要大量开发人员和应用等相关人员，因此，Web应用的开发方法和质量保证就非常重要和紧迫。

1998年，Yogesh Deshpande和Steve Hansen提出了Web工程的概念。Web工程作为一门新兴并快速发展的学科，提倡使用过程和系统的方法来开发高质量的Web应用。它使用合理的、科学的工程和管理原则，用严密和系统的方法来开发、发布和维护Web应用。相比国外，国内对Web工程的研究还有很大差距，系统地讲述Web工程的书籍也还处于空白。为此，各高校和公司纷纷开设课程和培训班。Web工程相关领域的研究、开发和教育也必然会不断地增长和深化。

为了适应Web工程的发展，本书旨在体现理论支撑和实践两方面的融合。在研究和实践之间，关键是建模和设计的理念。Web工程师不能只遵从设计说明。要构造出优秀的Web应用实则是科学与艺术的统一，这体现在对问题进行抽象，对它建模，并使用这些抽象设计出解决方案。

本书是为本科生Web工程课程而设计的，注重Web工程理论研究与实践的结合，使读者能够直接将所学的知识应用于要解决的实际问题。书中的例子业务背景学生相对比较熟悉，避免了对业务领域知识的学习工作量。本书也适用于介绍Web工程的概念和实践的研究生课程，还适合于那些期望进一步学习该领域相关知识的专业人员。

本书的主要特色是从Web应用的本质特征出发，以理论和实践有机地结合在一起的方式，介绍系统化开发Web应用系统的相关理论、技术和案例实践，既涉及产业实践需求，又包含学科前沿知识，符合教学实际。

本书并不详细介绍各种开发技术，比如JavaScript、HTML、基于.NET和Java EE等开发知识，这些具体技术在市面上已经有大量的书籍介绍。

本书重点强调Web工程，强调综合而系统化地开发高质量的Web应用的原理、方法、技术和工具。其内容涵盖Web应用的特征、Web项目需求工程、Web应用架构、Web应用建模、Web工程开发过程和Web项目管理、Web应用设计，Web应用开发和部署、测试、运行和维护，Web应用的性能、可用性、安全性，以及Web工程未来的发展趋势，案例贯穿全书。

本书所有章节的主要内容组织的共性是：首先概述每章内容在Web工程中的显性特点和作用，其次介绍每章内容所包含的Web工程目前的概念、方法、技术和工具，最后是总结和展望。

每章的主要内容如下。

第 1 章：Web 工程概述

作为全书的基础，本章从介绍 Web 的起源、发展、特性、原理技术等内容入手，分析 Web 应用的特性与分类，对 Web 工程进行了明确的定义，从多个方面区分了 Web 工程与传统软件工程。

第 2 章：Web 应用开发过程和方法

在分析 Web 应用开发过程的特点的基础上，介绍了常用的两种 Web 应用开发方法——极限编程和 RUP，并对它们对 Web 应用的适应性进行了详细的对比分析，然后着重介绍了它们对 Web 应用开发的应用实践。

第 3 章：Web 需求工程

在分析 Web 应用需求特性的基础上，从 Web 需求获取、Web 需求表示、Web 需求确认与验证、Web 需求分析工具等多个方面对 Web 需求进行了分析。

第 4 章：Web 应用建模

Web 应用建模是 Web 工程的一个重要内容，本章在分析 Web 应用建模特性的基础上，重点介绍了模型驱动开发、Web 应用建模的方法、功能需求建模、内容建模、超文本建模、表示建模和适应性建模等内容。

第 5 章：Web 应用架构

本章主要阐明模式、框架以及分析 Web 应用架构特性，重点介绍了模型驱动架构、层次架构、集成架构、面向数据的架构等内容。

第 6 章：Web 应用设计

Web 应用设计是 Web 开发的一个重要环节，本章在分析 Web 应用设计特性的基础上，重点介绍了交互设计、展示设计、内容设计、功能设计等内容。

第 7 章：Web 应用构建与部署

详细介绍了 Web 应用开发原则，分析了 Web 应用通信协议、Web 客户端开发技术、Web 服务器端技术、Web 应用开发框架、Web 应用构建工具以及 Web 应用部署等内容。

第 8 章：Web 应用测试

Web 测试是在 Web 提交给用户前的一个重要环节。本章从分析 Web 应用测试特性开始，对功能测试、内容测试、Web 页面测试、兼容性测试、性能测试、安全性测试、接口测试、Web 服务测试、测试工具，以及发展趋势多个方面进行了详细的分析与描述。

第 9 章：Web 应用的运行与维护

Web 应用部署完成并启动运行后，还需要进行大量的维护工作。维护是 Web 应用必不可少的一个环节。本章重点分析 Web 应用的运行和维护所面临的挑战，介绍如何推广营销以及维护 Web 应用。

第 10 章：Web 项目管理

Web 项目管理作为保障性活动面临着诸多挑战，本章涉及到的内容有 Web 项目人员管理、Web 应用项目计划、Web 项目风险管理、Web 项目配置管理等。

第 11 章：Web 应用的性能和可用性

本章涵盖 Web 应用的性能和可用性。在分析 Web 应用性能的基础上，阐述了提高 Web 性能的策略。又由于可用性对 Web 应用而言尤为重要，本章还介绍了 Web 应用的可

用性原则、可访问性、可用性和可访问性模式、移动可用性和 Web 可用性工程。

第 12 章：Web 应用的安全性

本章在分析 Web 应用安全性特性的基础上，介绍了 Web 应用所面临的安全性威胁和安全性防护策略，并从客户端、服务器端以及相互通信等方面描述 Web 应用如何防护所面临的安全性威胁。

第 13 章：Web 工程的发展

语义 Web 将是下一代 Web 的发展方向，也是 Web 工程后续发展的主要关注点。本章详细分析了 Web 技术的演化、Web 应用的发展演化，语义 Web 的体系结构以及语义 Web 应用领域，并介绍了 Web 工程的发展。

本书的编写得到了西安电子科技大学教材基金资助，大纲规划得到徐学洲教授、顾新教授、陈静玉等老师的大力指导和帮助，为本书的最后成功完成起到重大作用。

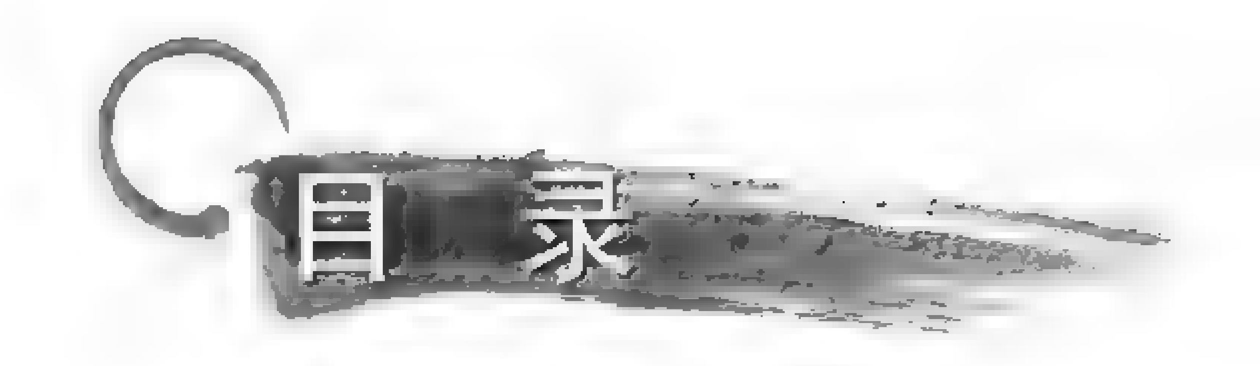
本书第 1 章、第 4 章、第 5 章、第 6 章、第 12 章由霍秋艳编写，第 2 章、第 9 章、第 10 章由陈静玉编写，第 3 章由邓岳编写，第 7 章、第 8 章、第 11 章由王黎明编写，由霍秋艳负责统稿。何昊收集并整理了大量的资料，闫兵参与了案例的设计，徐学洲审阅了全书。

感谢所有为本书的撰写提出过宝贵意见和大力支持的人。首先感谢所有参与本书撰写的作者，他们的专业知识、激情和努力，使得本书的撰写能顺利完成。感谢所有支持本书撰写的领导、同事、学生和朋友们，感谢他们对本书的成功撰写做出的贡献。感谢本书所参考的诸多讨论构建高质量 Web 应用的准则和技术的出版物以及 Web 资源的作者们。

作者力图反映 Web 工程这一新兴学科所涉及的基本原理和发展方向，并力图用通俗的语言讲述相关实践。但是限于作者水平，书中难免有错误与欠妥之处，恳请读者批评指正。

作 者

2011 年 2 月



目录

第 1 章 Web 工程概述 …… 1

1.1 Web 特性 …… 1

1.2 Web 应用 …… 3

1.2.1 Web 应用分类 …… 4

1.2.2 Web 应用特性 …… 6

1.3 Web 工程 …… 10

1.4 小结 …… 12

第 2 章 Web 应用开发过程和方法 …… 13

2.1 Web 应用开发过程的特性 …… 13

2.2 软件开发过程 …… 14

2.2.1 RUP …… 15

2.2.2 XP …… 19

2.2.3 RUP 与 XP 对 Web 应用的适应性 …… 23

2.3 定制基于 RUP 和 XP 的 Web 应用开发过程 …… 25

2.3.1 基于 RUP 和 XP 的 Web 应用开发过程 …… 25

2.3.2 敏捷 Web 应用开发过程 …… 33

2.4 总结与展望 …… 34

第 3 章 Web 需求工程 …… 35

3.1 Web 需求特性 …… 35

3.2 Web 需求获取 …… 38

3.2.1 需求准备 …… 38

3.2.2 需求获取方法 …… 39

3.2.3 需求获取原则 …… 40

3.2.4 敏捷需求获取 …… 42

3.3 Web 需求分析、表示与管理 …… 43

3.3.1 Web 需求分析 …… 43

3.3.2 Web 需求表示 …… 44

3.3.3 Web 需求管理 …… 46

3.4 Web 需求确认与验证 …… 49

3.5 Web 需求工具 …… 52

3.6 总结与展望 …… 55

第 4 章 Web 应用建模 …… 56

4.1 Web 应用建模特性 …… 56
4.1.1 层 …… 57
4.1.2 方面 …… 58
4.1.3 阶段 …… 58
4.1.4 适应性 …… 58
4.1.5 Web 应用建模的优点 …… 58
4.2 模型驱动开发 …… 59
4.3 Web 应用建模方法与工具 …… 60
4.3.1 UWE …… 61
4.3.2 WebML …… 64
4.3.3 HDM-lite …… 67
4.3.4 OOHDM …… 67
4.3.5 WebSA …… 69
4.3.6 其他方法 …… 69
4.3.7 小结 …… 73
4.4 功能需求建模 …… 73
4.4.1 绘制用例图 …… 74
4.4.2 绘制活动图 …… 76
4.5 内容建模 …… 76
4.5.1 静态建模 …… 77
4.5.2 动态建模 …… 77
4.6 超文本建模 …… 78
4.6.1 静态建模 …… 78
4.6.2 动态建模 …… 79
4.7 展示建模 …… 81
4.7.1 静态建模 …… 81
4.7.2 动态建模 …… 83
4.8 适应性建模 …… 84
4.8.1 静态建模 …… 85
4.8.2 动态建模 …… 85
4.9 总结与展望 …… 87

第 5 章 Web 应用架构 …… 88

5.1 Web 应用架构及其特性 …… 88
5.1.1 模式 …… 88
5.1.2 框架 …… 90

5.1.3 架构分类 …… 91
5.1.4 Web 应用架构特性 …… 91
5.2 模型驱动架构 …… 93
5.3 层次架构 …… 94
5.3.1 两层架构 …… 95
5.3.2 三层架构 …… 95
5.3.3 N 层架构 …… 97
5.4 集成架构 …… 98
5.4.1 门户 …… 99
5.4.2 EAI …… 99
5.4.3 SOA …… 100
5.5 面向数据的架构 …… 101
5.5.1 以数据库为中心的架构 …… 101
5.5.2 Web 文档管理架构 …… 101
5.5.3 流媒体数据的架构 …… 102
5.6 总结与展望 …… 104

第 6 章 Web 应用设计 …… 105

6.1 Web 应用设计特性 …… 105
6.2 交互设计 …… 108
6.2.1 用户交互 …… 108
6.2.2 用户页面组织 …… 109
6.2.3 导航设计 …… 110
6.2.4 复杂活动的会话设计 …… 110
6.2.5 交互设计原则 …… 111
6.3 展示设计 …… 114
6.3.1 Web 页面特性 …… 114
6.3.2 用户行为习惯 …… 117
6.3.3 页面布局设计 …… 117
6.3.4 页面元素设计 …… 119
6.3.5 美学设计 …… 122
6.3.6 展示设计原则 …… 123
6.4 内容设计 …… 125
6.4.1 信息设计方法 …… 126
6.4.2 信息架构 …… 126
6.4.3 组织内容 …… 127
6.4.4 访问信息 …… 128
6.5 功能设计 …… 129
6.5.1 集成 …… 129

6.5.2 分布式 Web 应用 …… 130
6.5.3 功能设计原则 …… 130
6.6 总结与展望 …… 131

第 7 章 Web 应用构建与部署 …… 133

7.1 Web 应用构建原则 …… 133
7.2 Web 应用通信协议 …… 135
7.3 Web 客户端技术 …… 137
7.4 Web 服务器端技术 …… 146
7.4.1 Web 应用服务器端开发技术 …… 146
7.4.2 中间件技术 …… 151
7.4.3 Web 服务 …… 153
7.5 Web 应用开发框架 …… 156
7.5.1 Java EE 开发框架 …… 156
7.5.2 .NET 框架 …… 159
7.5.3 Web 层开发框架 …… 160
7.5.4 Ruby 框架 …… 163
7.5.5 Python 框架 …… 163
7.5.6 Web 服务开发框架 …… 163
7.5.7 Web 应用开发框架的选择 …… 165
7.6 Web 应用构建工具 …… 165
7.7 Web 应用部署 …… 167
7.8 总结与展望 …… 169

第 8 章 Web 应用测试 …… 170

8.1 Web 应用测试特性 …… 170
8.2 Web 应用测试过程 …… 171
8.3 功能测试 …… 173
8.4 内容测试 …… 175
8.4.1 内容测试的目标 …… 176
8.4.2 验证动态内容 …… 176
8.5 Web 页面测试 …… 177
8.6 兼容性测试 …… 180
8.7 性能测试 …… 182
8.7.1 性能测试目标 …… 182
8.7.2 性能测试过程 …… 183
8.7.3 性能测试内容 …… 184
8.7.4 性能测试方法 …… 187
8.8 安全性测试 …… 188

8.9 接口测试 …… 190
8.10 Web服务测试 …… 191
8.11 测试工具 …… 192
8.12 总结与展望 …… 194

第9章 Web应用的运行和维护 …… 195

9.1 Web应用运行和维护的挑战 …… 195
9.2 推广营销 …… 197
9.2.1 网络广告 …… 197
9.2.2 搜索引擎营销 …… 198
9.2.3 病毒式营销 …… 202
9.2.4 Web 2.0推广 …… 204
9.3 内容维护 …… 205
9.4 Web使用挖掘 …… 206
9.5 总结与展望 …… 214

第10章 Web项目管理 …… 215

10.1 Web项目管理面临的挑战 …… 215
10.2 Web项目人员管理 …… 219
10.2.1 团队组织 …… 219
10.2.2 团队管理 …… 221
10.3 Web应用项目计划 …… 223
10.3.1 进度管理 …… 223
10.3.2 成本管理 …… 226
10.4 Web项目风险管理 …… 227
10.4.1 Web工程风险特性 …… 227
10.4.2 风险评估 …… 229
10.4.3 风险控制 …… 231
10.5 Web项目配置管理 …… 232
10.5.1 Web配置管理的内容 …… 232
10.5.2 配置管理的实施 …… 235
10.5.3 配置管理的工具 …… 236
10.6 总结与展望 …… 238

第11章 Web应用的性能和可用性 …… 239

11.1 Web应用性能 …… 239
11.1.1 Web应用性能分析 …… 239
11.1.2 Web应用性能提升策略 …… 241
11.2 Web应用可用性 …… 244

11.2.1 Web可用性原则 …… 245
11.2.2 可访问性 …… 253
11.2.3 可用性和可访问性模式 …… 254
11.2.4 移动可用性 …… 255
11.2.5 Web可用性工程 …… 256
11.3 总结与展望 …… 260

第12章 Web应用的安全性 …… 262

12.1 Web应用安全性特性 …… 262
12.2 Web应用安全威胁 …… 263
12.2.1 安全威胁种类 …… 263
12.2.2 安全漏洞 …… 265
12.3 安全性策略 …… 269
12.3.1 安全性相关技术 …… 269
12.3.2 安全生命周期体系 …… 272
12.4 客户端安全防护 …… 274
12.5 服务器端安全防护 …… 275
12.6 客户-服务器之间通信防护 …… 276
12.7 总结与展望 …… 279

第13章 Web工程的发展 …… 281

13.1 Web技术的演化 …… 281
13.1.1 Web 2.0 …… 281
13.1.2 Web 3.0 …… 283
13.1.3 Web 4.0、Web 5.0 …… 284
13.2 Web应用的发展演化 …… 285
13.2.1 新的Web应用种类 …… 285
13.2.2 新的计算模式 …… 287
13.2.3 多渠道访问 …… 289
13.2.4 Web操作系统 …… 290
13.3 语义Web …… 291
13.3.1 语义Web架构 …… 291
13.3.2 语义Web应用 …… 293
13.4 Web工程发展 …… 297
13.5 总结与展望 …… 298

参考文献 …… 299

第1章 Web工程概述

现代Web应用是全面而复杂的软件系统，因此，开发Web应用需要坚固的方法学的支持。Web工程是为了开发出高质量的Web应用，使用系统化、可度量的工程化方法来对Web应用进行需求定义、建模、实现、运行和维护。本章从Web特性出发，介绍Web应用的分类和特性，以及由其自身开发以及使用所产生的对Web工程的特殊需求。

1.1 Web特性

Web是Internet上的多媒体信息查询工具，是Internet上近些年才发展起来的服务，也是近些年来发展最快和目前使用最广泛的服务。正是因为有了Web工具，才使得近年来Internet迅速发展，用户数量飞速增长。

Web(万维网)起源于1989年，由欧洲粒子物理实验室(European Laboratory for Particle Physics，CERN)所发展出来的主从结构分布式超媒体系统。最早的网络构想要追溯到遥远的1980年Tim Berners-Lee构建的Enquire项目，这是一个类似于维基百科的超文本在线编辑数据库。尽管这与现在使用的万维网大不相同，但是它们确有许多相同的核心思想，甚至还包括一些Tim Berners-Lee的万维网之后的下一个项目语义网中的构想。

1991年8月6日，Tim Berners-Lee在alt.hypertext新闻组上贴出了万维网项目简介的文章。这一天也标志着Internet上万维网公共服务的首次亮相。Tim Berners-Lee的另一个才华横溢的突破是将超文本嫁接到Internet上。他在《编织网络》一书中，解释说他曾一再向这两种技术的使用者们建议将这两种技术结合，但是却没有任何人响应他的建议，最后他只好自己实施这个计划，发明了一个全球网络资源唯一认证的系统：统一资源标识符(Uniform Resource Identifier，URI)。

1994年10月，国际组织万维网联盟(World Wide Web Consortium，W3C)在麻省理工学院计算机科学实验室成立，专门致力于创建Web相关技术标准并促进Web向更深、更广发展。

长期以来，人们只是通过传统的媒体(如电视、报纸、杂志和广播等)获得信息。但随着计算机网络的发展，人们想要获取信息，已经不再满足于传统媒体那种单方面传输和获取的方式，而希望有一种主观的选择性。由于计算机网络的发展，信息的获取变得非常及时、迅速和便捷。通过Internet，人们只要使用简单的方法，就可以很迅速方便地取得丰富的信息资料。由于用户在通过Web浏览器访问信息资源的过程中，无须再关心一些技术性的细

节，而且用户界面非常友好，因而Web在Internet上一经推出就受到了用户的热烈欢迎，风靡全球，并迅速得到了爆炸性的发展。图1.1为Web站点数目随时间变化曲线图，到2010年12月达到2亿5千万个。现在，网络上提供各种类别的数据库系统，如文献期刊、产业信息、气象信息、论文检索等。

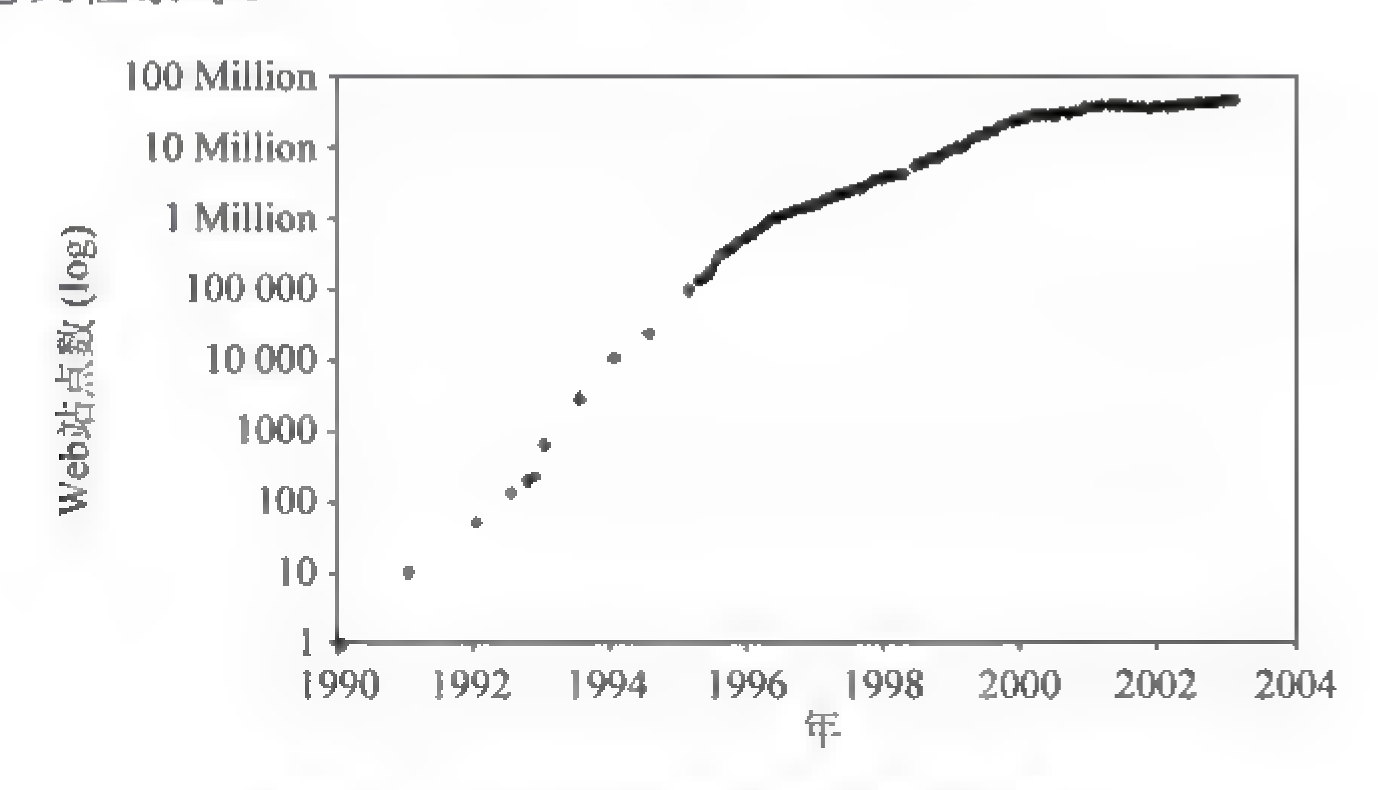


图1.1 Web站点数目随时间变化曲线

Internet采用超文本和超媒体的信息组织方式，将信息的链接扩展到整个Internet上。Web就是一种超文本信息系统。Web的一个主要的概念就是超文本链接，它使得文本不再像一本书一样是固定的、线性的，而是可以从一个位置跳到另外的位置，用户可以从中获取到更多更有用的信息，可以跳转到别的主题上。用户想要了解某一个主题的内容只要在这个主题上单击一下，就可以跳转到包含这一主题的其他文档上去。

Web和其他超文本系统有很多不同之处，主要表现在以下几个方面。

① Web上需要单向链接而不是双向链接，这使得任何人可以在资源拥有者不做任何动作的情况下链接该资源。和早期的网络系统相比，这一点对于减少实现网络服务器和网络浏览器的困难至关重要，但它的副作用是产生了坏链的慢性问题。

② Web不像某些应用软件如HyperCard，它不是私有的，这使得服务器和客户端能够独立地发展和扩展，而不受许可限制。

作为一门新兴的技术，Web的存在大大提高了信息的流通效率。一般而言，Web具有的特性有以下5点。

1）导航性

Web能够一直非常流行的一个很重要的原因就在于它可以在一个页面上同时显示色彩丰富的图形和文本。在Web之前，Internet上的信息只有文本形式。Web具备可以将图形、音频、视频等信息集合于一体的特性。同时，Web是非常易于导航的，只需要从一个链接跳到另一个链接，就可以在各页面、各站点之间进行浏览了。

2）平台无关性

无论用户的系统平台、浏览器是什么类型，用户都可以通过Internet访问Web，浏览Web对用户的系统平台没有任何限制。无论从Windows平台、UNIX平台、Linux平台还是其他的应用平台，用户都可以访问Web。对Web的访问是通过浏览器（Browser）实现的，如Netscape的Navigator，NCSA的Mosaic，Microsoft的Internet Explorer，Firefox，Google的Chrome以及国产的360安全浏览器、搜狗等浏览器。

3）分布式

Web 上大量的信息，包括图形、音频和视频信息会占用相当大的磁盘空间，甚至无法预知信息量的大小。对于 Web 搜索而言，没有必要把所有信息都放在一起，信息可以放在位于不同地方的 Web 应用上，搜索时只需要在浏览器中指明这个 Web 应用的 URL 就可以了。通过这种方法使得在物理上并不一定处于一个 Web 应用中的信息在逻辑上一体化，从而从用户的角度来看这些信息是一体的。

4）动态性

由于各 Web 应用的信息包含应用本身的信息，信息的提供者可以经常对 Web 应用上的信息进行更新。如某个协议的发展状况、公司的广告、国际国内新闻、WiKi 等，这些都需要尽量保证信息的及时性，所以 Web 应用上的信息是动态的，而且是经常更新的。这一点是由信息的提供者保证的。

5）交互性

从 Web 作为信息传播的工具的角度讲，可以将交互性定义为信息的交互、互动和反馈。Web 动态的特性表现在 Web 是交互的，交互性对 Web 应用越来越重要，人们在浏览网页信息的同时，也期望能够参与到提供信息的互动之中。

Web 的交互性首先表现在它的超链接上，用户的浏览顺序和所到站点完全由用户自己决定。另外通过表单的形式可以从服务器上获得动态的信息。用户通过填写表单可以向 Web 服务器提交请求，服务器可以根据用户的请求返回用户需要的相应信息。浏览用户和 Web 站点可以进行交互，例如在数字图书馆，输入账号信息后，用户就可以阅览或下载电子书；用户还可以在留言板发表对某些问题的看法和见解；也可以在网上聊天室里通过互联网进行信息交流；等等。这种交互性是传统媒体所不具备的，它极大地增加了用户访问 Web 应用的乐趣。Web 媒体为人们构建了一个新型的并有无限乐趣的交流平台。

Web 的这些特性，使得将越来越多的传统应用软件制作成基于 Web 的系统变为可能，广阔的市场前景使得 Web 应用的发展快速而高效。

1.2 Web 应用

Web 应用是指那些用户界面驻留在 Web 浏览器中的任意应用程序，它基于 Web 技术和 W3C 标准，通过一个用户界面（Web 浏览器或支持 Web 技术进行访问的可视化界面）提供 Web 特定的资源，例如内容和服务等。

Web 应用是由动态脚本、编译过的代码等组合而成。它通常架设在 Web 服务器上，用户在 Web 浏览器上发送请求，这些请求使用 HTTP 协议，经过因特网和企业的 Web 应用交互，由 Web 应用和企业后台的数据库及其他动态内容通信。它是当前基于 Web 信息系统的主要表现形式，其特征是通过 Web 浏览器或 HTTP 用户代理访问实现服务。它一般由瘦客户端（Web 浏览器）、表达层（Web 服务器）、应用层（Web 应用服务器）以及数据服务层（数据库）组成。它所承担的任务不仅是简单的基于 Web 的数据发布，也包括了信息系统的构建和复杂的应用逻辑。

Web 应用可以是基于.NET 或者 Java 的程序、Web 服务、SOA 组件、混合的应用程序或者通过独立的业务功能编排的业务流程，也可以是面向客户的应用程序或者是后台的应

用程序，在特定的环境中能够以任何数量存在的这类应用程序。

Web站点和Web应用是同一事物通过不同视角观察的结果。Web应用是Web站点从外在功能角度的概括，Web站点是Web应用的实际存在形式和实现技术。Web站点分为静态Web站点和动态Web站点。静态Web站点的全部页面内容都来自事先生成的文件。动态Web站点的一部分页面内容通过服务器端的即时计算得到，即所谓的动态页面(Dynamic Web Page)。

1.2.1 Web应用分类

Web应用被分成运行浏览器的客户端与运行Web服务器的服务端。客户端负责将页面呈现给用户，服务端负责业务逻辑的处理与Web页面的构建。除了预定义的HTML页面是静态的之外，页面的构造过程完全是动态变化的过程，动态产生的HTML页面是服务器进行某些处理的结果。

Web应用的范围和复杂性千差万别，从小规模应用到大规模分布在Internet、Intranet和Extranet上的企业应用。现在的Web应用提供了巨大的各种各样的功能并且有不同的特性和需求，能够根据不同的方式分类，但是却没有一种很平常和广泛被接受的分类方式。在理解它们的需求和为了开发和部署Web应用的基础上，将Web的应用分为以下9类。

1) 以文档为中心的Web应用

以文档为中心的Web应用是Web应用的先驱，制作好的页面被存放在Web服务器上，例如静态的HTML文档会作为响应的请求被发送给客户端。这些Web页面经常需要使用工具手工更新。这类Web应用需求经常变更，或者大量的页面需要修改，导致耗资巨大而且信息经常过时。这类Web应用的一个主要优点是简单、稳定，并且响应时间短。

2) 交互式Web应用

随着CGI(Common Gateway Interface，公共网关接口)和HTML表单的引入，交互式Web应用出现了，它通过使用下拉框等控件与用户交互。Web页面和链接也可以根据用户的输入动态产生，典型的例子就是虚拟展览、新闻类站点等。

3) 事务型Web应用

这类Web应用提供给用户更多的交互性，使得用户具有更大的主动性。它不仅可以让用户与Web应用的交互以只读的方式进行，还可以提供给用户进行基本的内容更新。这类系统一般包括网上银行、在线购物和客户预定系统等。

4) 基于工作流的Web应用

基于工作流的Web应用允许处理不同公司、公众人物、私人之间的工作流，需要有特定的自动化处理和操作的结构。这种类型的应用的一个重要驱动力是Web服务保证了互操作性。而服务的复杂性、涉及到的公司的自治性以及工作流的健壮性和灵活性就成了这类应用面临的主要挑战。这类Web应用包括如电子商务中的B2B(Business to Business)解决方案、电子政务、医疗系统等需要工作流支持的应用。

5) 协作型Web应用

协作型Web应用用于非结构化的操作来达到合作的目的，如群件(Groupware)系统。协作型Web应用提供共享信息和工作空间(例如维基百科等)，用于协作的用户之间进行沟通以产生、编辑、管理共享信息，也可用于记录日志(如在网络日志中记录和编辑)，也可用于一些会议或者聊天室、共享日历、电子学习平台等。

6）面向门户的 Web 应用

面向门户的 Web 应用提供了一种访问入口来访问不同的、异构的信息和服务，如微软和 Netscape 等浏览器厂商，如雅虎等搜索引擎，如 AOL 等在线服务或者其他一些需要使用到门户的企业，会提供一个用户访问 Web 的中央入口，就是门户。另外，还有商业门户，提供给业务合作伙伴通过局域网和外联网来访问不同的资源；市场门户，例如网上购物商城，被进行了横向或纵向分割，横向即 B2C 和 B2B 市场，纵向分割即把供应商和制造商各分在一部分；社区门户则针对特定用户群。

7）普适 Web 应用

普适 Web 应用在任何时候、任何地点为各种设备提供个性化服务，因此普适访问变得更加容易。例如，用户在移动设备上输入当日 11 时和 13 时之间营业的餐厅，然后显示出当日的菜单。对这种应用而言，考虑移动设备的特性（如带宽、屏幕大小、内存、软件的不成熟等内容）和 Web 应用的使用环境是非常重要的，需要能够根据用户当前的情境进行调整。已有的这类 Web 应用经常只能支持某方面的内容，比如个性化服务、位置感知服务和（或）跨平台等。

8）社会网络

最初 Web 的特点是匿名性，现在出现了越来越多的社会网络（Social Web），用户需要给具有相似心情的社团提供自己的身份信息。例如，网络日志或协作系统不仅用于发现相似兴趣的内容目标，而且还用于发现同类应用下具有相似需求的人。

9）语义 Web

目前，通信、信息技术、多媒体、教育和娱乐、安全等企事业的系统发展，尤其是不断的聚合，不久的将来，都将会成为普适应用。现有 Internet 中的页面功能单调、搜索引擎智能化程度低的问题导致用户的操作变得越来越麻烦。语义 Web 就是为适应这些发展和解决这些问题而提出的。在语义 Web 中，信息不仅是提供给人们阅读理解，也可以提供给机器阅读理解。这样，在 Web 上管理、链接、重用和发现知识就会非常容易，比如推荐系统。通过在语义级的互操作和任务的自动化，相信 Web 会越来越普适，也因此和人们的日常生活的联系更加紧密。

一般而言，Internet 上的 Web 应用都隶属上述的某一类，或者会跨越几个种类，如幸福密码网（www.xmima.com）既包含以电子杂志、音频、视频等媒体为中心的功能，又包含团体测试等事务型功能，还提供咨询平台这类用户之间的协作功能。Web 应用也会在后续从一种发展到另一种，不管是哪一种，分清待开发的 Web 应用的类别，有助于系统化和工程化地开发与维护高质量的 Web 应用。

另外，从发展的角度看，Web 的发展将不同的学科如多媒体、信息科学、信息和通信技术汇集在一起，以提供在任何地方任何时间以不同的访问方式创建、维护、共享和使用不同信息。Web 的发展可以从不同方面不同视图进行表述，本节不展开讨论。为了更好地从 Web 设计的角度表述 Web 系统和应用，这里将根据创建的特性分为如图 1.2 所示的 5 类。

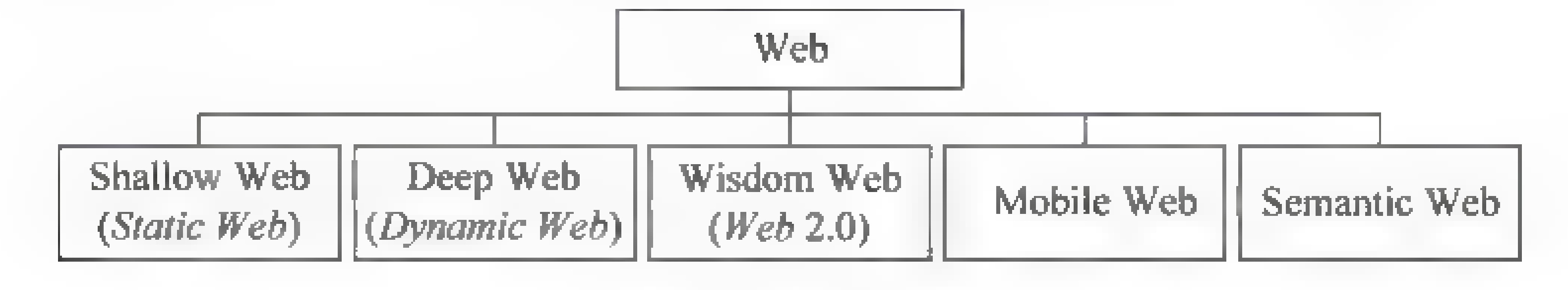


图 1.2 Web 分类

(1) Shallow Web 和 Deep Web。Shallow Web，也称为静态 Web，通过静态 HTML 页面提供信息共享。而后，Web 页面逐渐变为动态创建，即 Deep Web，也称为动态 Web。动态 Web 使得用户必须自己访问网站并发起请求，才能获取到信息。Shallow Web 和 Deep Web 站点中用户交互性低，现在都被 Web 1.0 所涵盖。

(2) Web 2.0：Web 的新面孔。Web 2.0 是一系列技术、业务策略和社会趋势的集合，具有高交互性、动态性。

最近几年出现了一类新的 Web 应用(称为 Web 2.0，或面向服务的应用)。这些应用使得人们可以在线协作和共享信息，如 myspace.com、youtube.com 和 Wikipedia 等。新一代的 Web 也称为 Wisdom Web(智慧 Web)、以人为中心的 Web 或读/写 Web，其核心思想是用户参与创作，通过一个富用户界面集成多种服务。而且，AJAX、Flex、Ruby、Blog、WiKi 和社会化书签等技术使得 Web 很快变得更动态和具有更高的交互性，由此又引出了另一类应用——RIA。

RIA(Rich Internet Application，富互联网应用)是一种基于 Web 的应用，具有和传统桌面软件类似的功能和特征，运行于 Web 浏览器，而不需要安装其他软件。RIA 于 2002 年在 Macromedia 的白皮书中提出，展现了浏览器从静态请求响应页面发展成动态、异步页面。带宽、客户需求、应用技术包括 Web 2.0 等，驱使和促进了 RIA 的普及与发展。RIA 在交互性和可用性方面有非常大的提升，提供更丰富的用户体验和更多的优势。典型的 RIA 框架如 Adobe 的 Flex、AJAX，应用如 Google 的 Earth、Gmail、Finance。

很多企业因 RIA 的特点，而将其应用于用户交互任务中，使用如 AJAX、Flex 和 Web 服务等技术进行实现。然而，使用各种时尚的技术构建 Web 应用并不能确保有更好的用户体验。要使 RIA 具有实用价值，开发人员必须了解用户的真正需求以理解使用哪些合适的 RIA 技术，实现结构化测试技术来验证 RIA 的设计。

(3) Mobile Web。移动计算和无线通信技术的发展，使得移动手持设备成为另一条访问 Web 的重要渠道，而且在进一步普及。为了支持移动设备的访问，很多 Web 应用程序也大力支持移动和无线访问，称为移动 Web，而且 WWW 组织也建立了 Mobile Web。这类移动应用提供对用户特性方面的支持，如位置感知服务、上下文感知能力和个性化。

(4) Semantic Web(语义 Web)。目前的 Web 应用程序中，信息大多采用自然语言描述。人类可以很容易地处理这些信息，但是计算机却不能。语义 Web 就是为了解决这种计算机理解自然语言而进行的扩展，其中信息的意义被很好地定义，让机器能够处理全局信息，使得人与计算机之间可以进行互操作。

1.2.2 Web 应用特性

Web 应用和传统软件应用相比，具有很大的不同。因此，理解 Web 应用特性是更好地设计 Web 应用的重要基础。

和传统软件相比，Web 应用的独有特性，使得其开发更加困难，甚至更具挑战性。Web 应用具有开发时间短、初始需求不明确等显著特点。以 ISO/IEC9126-1 标准对软件特性分类为基础，将 Web 应用的特性分为产品、使用、开发以及演化 4 个方面。产品必须是可适应的，在使用时应当考虑到新的上下文信息，开发面临着不断的变更。这些特性将会在后续多个章节中不断体现，下面从这 4 个方面加以描述。

1. 产品特性

传统软件工程强调的是系统功能，除了系统帮助等一些文档以外，其他的几乎就是一些数据交互功能，是面向功能的。而Web应用是内容驱动的，强调的是内容（信息）、超文本结构（导航结构）和展示（用户界面，包含静态或动态页面及其视觉和感觉等艺术表现），按照面向对象的表述方式，这些部分不仅是结构的或静态两个方面，而且是行为和动态性两个方面。

1）内容方面

其产生、提交、集成和更新与开发Web应用软件本身同等重要，强调的是内容的含量。Web应用之所以被广泛使用，很重要的原因就是其提供了及时有效的内容信息。因此，很多Web应用开发人员在充当程序员的同时，也充当着内容创作者的角色。这主要体现在以文档为中心和多媒体特性以及用户对内容质量的高要求，即内容结构的不同，可以表示为表格、文本、图片、动画、音频或视频等。这些内容提供来为特定用户群体服务，比如航班信息或宾馆空闲可预订的房间信息等，需要动态实时地更新相应的信息，可用性就非常重要，因此，需求定义时就需要考虑内容质量，也需要对其进行很好的测试。

以新闻Web应用为例，其内容更新频率和用户对内容的时事质量要求都会非常高。Web作为展示视频、音频、文字等内容的载体，比传统的电视、广播、报纸等，内容和更新频率和质量要求更高。比如个性化，用户可以根据自己的兴趣爱好进行定制，可以通过感知用户位置提供基于位置的服务，当出差到达另一个城市时，手机上提供当地的新闻信息。已有内容无法直接适应这些个性化的应用，必须进行重新开发。如网上购物系统，其内容的质量非常重要，价格、商品数量等信息如果不正确或者已经过时，将会使得交易失败，等等。再如幸福密码网上的电子杂志等内容，其领域知识和展示方式都要体现为了幸福。

不管Web应用在何时何地被使用，要让用户接受，内容的质量至关重要。要同时保证大量内容的质量以及不断且及时地得到更新，是现有以内容为中心的Web应用所面临的最大挑战。

2）超文本方面

有多种不同的用于表达信息的超文本模型，Web给出了非常简单的一种，可以把超文本模型的基本元素定义为结点（Nodes）、链接（Links）和锚（Anchors）。其中，结点为最基本的信息单元，比如HTML文档格式的Web页面，可以通过URL找到；链接是结点之间的通路，这些通路通常是单向的，而且意义表达也不够清楚；锚是一个结点内容范围内的区域，是链接的源或目标，只能存在于HTML文档，比如文本中的单词序列或绘图中的图形对象。

超文本的主要特性是它的非线性本质特性和从而导致的用户容易产生的“迷失”或感知负担，这成为Web应用的非常特殊的特性。超文本是Web应用和传统软件之间最大的一个区别。用户可以在自己感兴趣的和熟悉的信息空间中随意穿梭。锚和链接不仅可以由创作者自己预先定义，也有根据用户行为模式动态生成的。Web应用强调动态性，即服务器端的应用程序或脚本通过响应用户的请求产生Web页面，产生的Web页面可能包括客户端的脚本，如JavaScript、VBScript等。每个脚本都有自己的任务，产生的Web页面连同客户端的脚本又会产生一些无法预知的行为。因此，对于创作者而言，最大的挑战就是如何避

免无向性和用户的感知负担。

在Web应用开发中，处理超文本的无向性和用户的感知负担非常重要。无向性容易使人们在非线性的文档中"迷失"，感知负担即需要用户同时记住多条访问路径或多个任务。这种情况下，网站地图、关键字搜索、显示访问路径和访问时间等，有助于使用户清楚地保持自己的路线；使用有意义的链接和智能链接名有助于减轻用户的感知负担；超文本建模时使用设计模式也有助于解决这个问题。

3）展示方面

Web应用展示的最大特点是用户界面既包含艺术特性，又需要能够自我解释。虽然Web应用的界面设计也逐渐被软件工程的人机交互研究领域所涵盖，但与传统软件界面以"简单为美"的原则相比，Web应用则在多种程度上与多媒体结合，强调颜色搭配、动画的展示等艺术性，以及界面在没有帮助文档的情况下的自我解释性。Web应用关注视觉和感觉，强调感官舒服。一个展示视觉效果不好的Web应用，难以维持其竞争力，用户很容易就流失向其他同类Web应用。Web应用的导航和交互行为在整个应用中应该保持一致，这样用户才能快速熟悉此应用的使用。易于使用的Web应用有助于吸引和留住用户。

2. 使用特性

传统软件的用户可以圈定在某个范围之内，可以根据这个群体的特征设计用户界面。Web应用的用户的数量、文化背景、所使用的设备软硬件、访问位置和时间、访问频率等多种多样，而且无法被预知，并且自发访问。开发人员无法预先知道和影响这些上下文信息。Web应用必须不断地适应各类用户和各种上下文情况下的使用，比如内容、超文本和展示都应根据上下文而进行调整。这些上下文可以分为社会上下文、技术上下文和自然上下文。

1）社会上下文

社会上下文主要指用户特定的方面。用户具有不同的文化背景，又自发访问，想来则来，想走则走。而Web应用并非是为某一类特定用户开发的，并且在开发Web应用时，难以预知用户的技能、知识和特性，以及何时访问。因此，在开发阶段就需要设想一些用户上下文，并提供相应的个性化支持，比如为一些普通用户提供内容方面的适应性，为新客户提供使用指导，为有视觉颜色障碍的用户提供不同字体大小，等等。通常情况下，个性化服务需要用户设置自己的偏好，比如某购物网站上支付方式信息。

2）技术上下文

技术上下文主要指网络连接的服务质量，以及访问Web应用的软件环境（即多平台交付）的服务质量。在技术上，Web应用主要是基于客户服务器的架构原理，而且是瘦客户端（用浏览器）通过Internet访问服务器。传统软件的网络基础设施在软件开发阶段一般都已经明确，而Web应用开发只能在开发阶段考虑多种情况，即需要考虑带宽、可靠性和连接稳定性等传输介质，比如，在网速慢的情况下，应该调整视频文件的分辨率以改变传输数据量，以提高视频的流畅性。Web应用通常不是仅只支持一种特定的设备，这就需要考虑多平台交付的需求，需要考虑如移动设备的屏幕、内存、所装软件等方面。比如，用户使用不同的浏览器，因其功能性、限制和对标准的支持的不同，是Web应用面临的一个重大挑战，而用户可能对浏览器做出的不同配置（如对Cookies和缓存的配置、Java Applet访问权限等），都会对性能、事务功能、交互能力等有很大影响。

3）自然上下文

自然上下文主要指访问的位置和时间，这对Web应用的全球可用性非常重要。比如要提供位置服务，物理位置就很重要，而位置服务的测试则面临着非常大的挑战。Web应用的全球性更加强调安全性，Web应用一旦交付，全球范围就可以访问，这就需要阻止可能来自非法用户以及黑客的入侵、攻击以及隐私和机密数据的泄漏，必须确保Web应用的安全性。Web应用对时间方面的考虑使得时间感知服务（如日程安排）越来越普及。

3. 开发特性

Web应用开发使艺术、技术和科学比通常意义上的软件开发在更大范围内结合。开发Web应用的团队在社团开发、多学科专业技能方面比传统的软件开发所需要的更加广泛，人员类型更多。Web应用具有异构性和不成熟性。由于Web应用采用开放性标准，HTML、XML、JavaScript、VBScript、CSS、JSP、ASP和第三方构件等技术都可以无缝地应用于Web应用。这些技术配置在不同的平台上，提供了操作上比较松散的服务，但是这些标准、技术和平台等都在不断更新，如出现AJAX等异步技术，使得页面只需局部更新，就能达到页面内容更新的目的，甚至HTML5也会在不久的将来成为更多厂商支持的标准，这样RIA的实现将更加方便。在这种情况下，开发人员相对也就显得更年轻，更没经验。

Web应用经常面临初始需求不明确，而且必须在短期内开发完成的情况，所以，多个功能可能在开发过程中并行进行，比如，认证授权、搜索功能和新闻模块等同时由多个开发人员（或小组）进行开发。因此，Web应用开发过程框架需要支持灵活性和并行开发特性，并且也受其影响。

另外，某个Web应用经常会作为更大应用的一个组件，会涉及技术、内容以及组织等多方面的内部或外部集成。内部集成发生得最为频繁，Web应用需要和内部遗留系统集成，比如产品目录（即内容）需要通过Web应用进行访问。外部集成涉及到内容和服务的集成。某些方面和数据集成类似，但也有Web应用特有的集成需求。

4. 演化方面的特性

传统软件的演化是在有计划地进行版本升级。但Web应用不论在产品本身，还是其使用或其开发，都日新月异。新的需求和特征不断出现或变更，内容、使用和用户不断增加，技术和标准不断发展，开发和生存期越来越短，问世速度越来越快，变更周期已经不是几天或者几周，甚至会是几个小时，安全性要求也越来越高。因此Web应用的开发无法像传统软件开发过程一样，强调需求的分析和规格说明，而更多的挑战工作来自于运行和维护。

上述4个方面的特性说明，现代的Web应用越来越成为一个庞大的集成方案，它不再仅仅是单一的开发工作了，而是需要考虑到不同的操作平台、应用服务器、数据库、编程语言、传输介质、访问渠道、协作、安全性等。各种支持软件和技术在不断发展演化中，超越窄带的互联网，还可能涉及到带宽所带来的变动，或是增加与无线移动的接口。如果缺乏严格的过程，不进行系统化的分析、建模、设计、运行维护以及项目管理，很难期望能开发出一套跨平台、健壮、易扩展和易升级的高质量Web应用。

而且，随着Web应用变得越来越复杂，一个项目的失败将可能导致很多问题，人们对Web和Internet的信心可能会无法挽救地动摇，从而引起Web危机。并且，Web危机可能

会比软件开发人员所面对的软件危机更加严重，更加广泛。因此，人们迫切地需要使用系统化和工程化的方法来进行Web应用的开发。

1.3 Web工程

20世纪90年代末，由于Web应用不断大型化，功能复杂化，引发了Web危机(Web Crisis)，与出现在20世纪60年代的软件危机有惊人的相同之处，即初期开发规模小，以个体行为为主，转向大型系统开发后项目需求日趋复杂，但仍然沿用个体化开发方式，对开发人员个人素质过于依赖，导致开发质量下降，成本升高，维护难度越来越大，失败项目屡见不鲜。在20世纪60年代，开发软件被等同于编写程序；而在20世纪90年代末，开发Web应用则被等同于编写Web页面。Web危机源于超文本技术本身的缺陷，正是在Web应用的开发过程中缺乏工程化的手段，Web危机愈演愈烈。

1. Web应用与传统软件开发的区别

传统的软件工程方法虽然可以在Web应用开发中发挥一定作用，但由于Web应用及其开发过程的特性，在开发中难以将全部软件工程的理论、规则和经验加以运用。大量开发经验也表明，结构化开发、建模、面向对象、快速原型法等常见的软件工程手段在实践中并不能像期望的那样发挥作用，因此，迫切需要更适合的过程和方法、工具来开发、发布和评估Web应用。

虽然Web工程包含了程序设计和软件开发的内容，也采用了一些软件工程的原理，但是，Web应用的开发与软件开发有很大不同，同样，Web工程也不同于传统的软件工程。传统的软件工程在Web开发中也存在其自身的局限性，主要表现在以下几个方面。

首先，Web应用本身处于持续变更中，不完全适合螺旋模型等软件生存期模型描述。

传统的应用软件在交付用户后便进入一个相对稳定的阶段，即软件的维护。在这个阶段中，客户所持有的软件代码保持不变，即使客户在使用中发现各种问题和缺陷，开发者也只会收集这些反馈并在内部对程序进行小的修改，直至这类小的修改积累到一定程度才以一个“补丁”或者一个新版本的形式交付给客户，呈现出明显的“发布—反馈—更新—再发布”的阶段性过程，而Web应用的生命周期则没有明显的阶段划分。一般来说，客户发现在Web应用中的各类问题后总是要求开发人员立即进行修改。另一方面，内容增加导致新的页面不断被添加到系统中也使得Web应用处于不断变更的过程中。因此，Web应用开发中的生命周期界限模糊，缺乏版本控制的基础。

其次，Web应用开发的需求分析比传统软件开发的需求分析更困难。

需求分析在传统软件开发中占有非常重要的地位，能否明确了解客户的需要是项目能否顺利进行的关键因素。客户需求的中途改变会对项目的质量管理、进度控制及成本控制产生非常严重的负面影响。因此，传统软件工程非常重视需求分析的质量，并已经总结出一套有效的需求分析的方法与规则，例如用例(Use Case)图和快速原型(Rapid Prototype)法等。然而这些方法在Web应用开发中效果不佳，一方面，和传统软件系统的客户需求相比，Web应用的客户需求中涉及界面和外观的比例大大增加，功能性需求和外观需求往往相互交织。而传统的需求描述手段如用例图只适合进行功能性的描述，对外观缺乏描述能力。

目前Web应用中的前端部分的需求以自然语言描述附以非标准化的草图居多，大大降低了需求分析及其相关文档的作用。另一方面，Web应用开发的客户对需求的概念更模糊，往往缺乏一个完整的构想和期望，其需求比传统软件系统的客户更容易随着时间而改变。而且，由于Web应用开发的技术门槛相对较低，也使得客户提出的需求更具有随意性。部分对Web应用开发的技术有一定了解的客户更容易忽视需求变更可能损害项目质量的警告，使得在项目开发初期所做的需求分析形同虚设。这些都导致目前Web应用开发的需求分析中缺乏一种真正有效而规范的手段。

此外，Web页面属于一类特殊的人机界面(Human-Machine Interface)。

虽然目前Web页面中出现了越来越多传统软件中的用户界面元素(User Interface Element)，如按钮、下拉框等，但Web页面的主体仍然是超文本文档，Web应用中最主要的界面元素仍然是超链接。通过单向超链接相互关联的超文本文档群在结构特性上也与传统的应用程序大大不同。这些差异使得对Web页面的用户使用习惯需要重新进行研究，例如已有一些研究着重探讨页面导航(Navigation)对用户使用感受的影响。同时，从传统应用程序归纳出来的一些界面易用性准则也不再全部适用于Web应用。

2. Web工程及其相关领域

为了保证Web应用开发的高效率、高质量和低风险，有必要对整个开发过程进行管理和规范。计算机科学关注硬件和系统软件，以获得优化和可靠的性能。软件工程关注大规模、基于团队的项目，信息系统来自数据处理，关注企业级的信息。Web工程就是实现工程化开发基于Web的系统。

澳大利亚Yogesh Deshpande和Steve Hansen等科学家认为信息产业界应该把Web工程看做是一门崭新的学科。Yogesh Deshpande和Steve Hansen提出Web工程的概念：Web工程是一门关于建立科学的、工程的、管理的原则，采用系统的、严密的方法来开发、实施和维护Web应用的学科，而非软件工程的一个分支，旨在满足Web应用开发的各种特殊需求。

目前，大家比较能够接受的Web工程的定义为：Web工程作为一门新兴的学科，提倡使用一个过程和系统的方法来开发高质量的Web应用，它使用合理的、科学的工程和管理原则，用严密的和系统的方法(如概念、方法、技术和工具)来分析、设计、实现、测试、运行和维护高质量的Web应用。

Web工程的主要研究对象是运用Web相关技术开发的Web应用，尤其是复杂的Web应用。类似于软件工程，Web工程主要研究内容是Web应用的整个生命周期内的系统化方法以及非技术层面的问题，其主要研究目标是确保和提高Web应用开发的质量、性能和效率。Web应用本身也是一种应用软件，所以一些软件工程中的原理、概念和方法可以应用于Web工程中。和传统软件工程类似，其基本原理可以描述如下。

① 清晰地定义目标和需求。

② 系统地开发Web应用的阶段。

③ 严谨的计划。

④ 持续审计和评估开发过程。

Web工程和软件工程类似，并非一次性事件，而是跨越Web应用整个生命周期的过

程。Web工程使得迭代和有计划地开发Web应用以及持续演化成为可能，不仅可以降低开发和维护的费用和风险，还可以改进质量并且可以度量每个阶段的产出。

Web工程涉及多领域的相互协作，不但来自计算机科学、软件工程和信息系统，而且来自其他学科。Web工程目前已经成为一个多领域交叉的研究方向，如图1.3所示，涉及人机交互、信息检索、计算机图形学、信息工程、建模以及项目管理学等多方面课题。它代表了信息技术演化的方向。

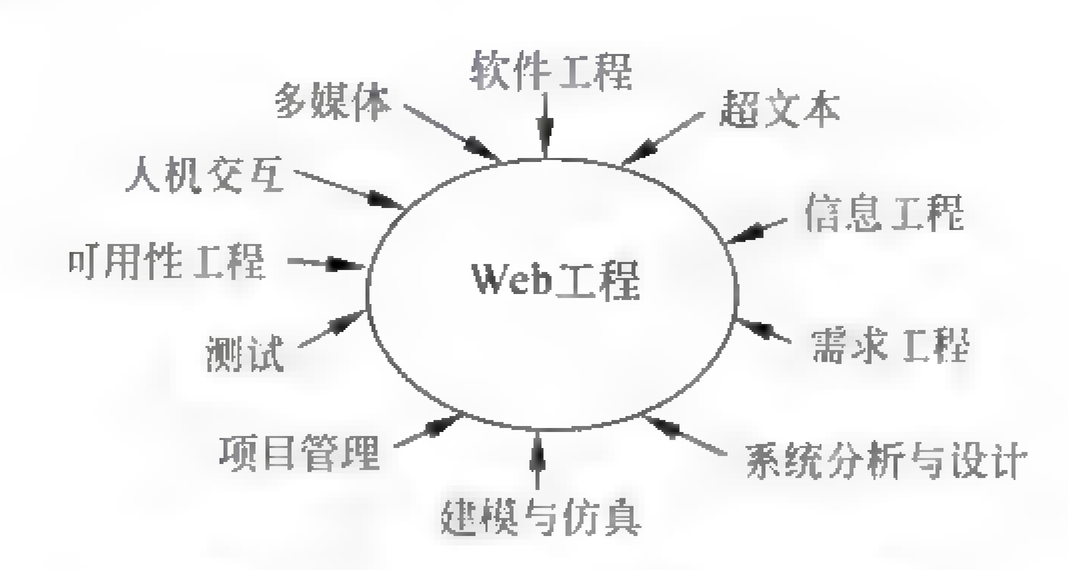


图1.3 Web工程涉及多个领域知识

Web工程是一个快速增长的领域，需要使用更加前沿的方法，而不是仅仅应用以前存在的方法和被证明了的开发实践。越来越多的研究者，致力于研究与Web工程有关的概念、技术、方法、工具和过程等，这也成为Web工程作为与软件工程相独立的一门学科发展的重要方面。

1.4 小结

Web的导航性、平台无关性、分布性、动态性和交互性等特性，使得Web应用具有其自身特性和分类。Web工程是为了开发高质量的Web应用，使用工程化、系统化的开发方法（如概念、方法、技术和工具），进行Web应用的开发。

本章首先分析了Web的特性，引入了Web中的相关概念，然后对Web应用进行了分析，给出Web应用的分类与特性，最后根据Web应用与传统应用软件的不同以及在实际的Web应用开发过程中出现的问题，给出了Web工程的定义，指明Web工程所涉及的不同学科知识。本章是所有后续章节的基础。

第2章 Web应用开发过程和方法

面对Web应用具有的业务逻辑复杂、业务流程更新速度快和运行于多个平台等基本要求，以及Web应用的特性，如果缺乏严格的过程控制，那么在进行Web应用开发、发布、运行和维护等过程中，可能就会碰到一些严重的问题。在Web应用的开发中，需要考虑如何定义一个适合的开发过程，即描述Web应用开发过程不同时期应有什么角色、应采取什么行为、经过几个工作流、应得到什么产品或达到什么目标。

在Web应用开发中参照软件开发的方法学和软件工程的思想是可行的。但是与传统的软件相比，Web应用具有网络密集、内容驱动、持续演进的特点，而Web应用的开发又具有即时性、安全性、追求页面美观等特点，使得Web应用开发过程和传统的软件开发过程有所不同。

2.1 Web应用开发过程的特性

与传统的软件开发相比，Web应用短时间交付，竞争激烈，注重用户界面，强调质量属性，客户认知高度可变，再加上它的细粒度的演化与维护、开放式模块化结构、快速的技术改变，导致开发技术涉及的知识面广，更新速度快。从整体开发过程看，Web应用呈现出以下一些有别于传统软件开发过程的特点。

(1) 开发周期短。Web应用所面对的竞争压力，使其开发周期一般情况下都比较短，通常不会超过6个月，平均不到3个月。Web应用短的开发周期成为Web应用开发过程中必须首要考虑的需求。

(2) 需求变更频繁。在开发过程中，Web应用的需求会经常慢慢浮现出来，或者因为内容和技术的变化而受到影响。而且，开发人员经常处理未知领域的业务逻辑，业务需求的认识会随着开发人员对项目以及业务的深入理解而发生改变。为了赢得市场集成的胜利，变化的市场环境所可能产生的新需求，必须快速加以集成，以获得竞争优势。

(3) 开发技术不断演化。随着Web应用在各行各业的不断普及，很多Web应用往往不单只是面向专业用户，而是要能方便地支持各类用户的操作需求，提高用户的使用体验。尽可能采用新颖、高效、美观的界面和交互技术，是大多数Web应用的核心要求之一，而这些界面和交互技术有些可能会对整个应用的架构和设计开发产生影响，为项目开发带来不确定性。

(4) 并行开发不同版本。Web应用开发市场竞争激烈，这导致竞争者都试图缩短发布

周期，用最短的时间开发出满足需要的 Web 应用。在这种时间压力下，只有重叠开发或并行开发才能使一个完整的 Web 应用按期完成或提前完成。这也意味着对于不同的发布版本设计、执行以及质量评估阶段的方法活动必须同时有效。

(5) 重用和集成。为了应对时间压力，提高 Web 应用的开发效率和开发质量，很多情况下都不会盲目地进行开发工作，而是采取基于构件的开发方式，因此，对于 Web 应用开发过程而言，支持重用和集成不同的构件就显得非常重要了。

(6) 适应 Web 应用的复杂性程度。在 Web 应用开发过程中，一般情况下短开发周期比很多质量需求优先级更高。很多软件的质量属性经常被忽视如可伸缩性和可维护性，导致在后续版本中，需要通过替换构件或开发新构件以弥补开发阶段早期忽略的质量特性，这就需要更详细的开发计划和文档。

从前面的讨论中可以看出，Web 应用开发过程需要做到以下几点。

(1) 基于迭代思想，重视系统的快速开发和不断演化，降低在一个增量上的开发风险。

(2) 强调原型开发，并作为开发过程模型的重要组成部分。

(3) 强调开发过程中各个阶段的追溯、调整和反馈。

2.2 软件开发过程

为了应对软件的日益复杂程度、漫长的开发周期以及用户对软件经常不满意等方面的状况，软件工程强调使用生命周期方法和各种分析及设计技术来开发软件，并由此产生了软件过程模型。软件过程模型建议用一定的流程将各个环节连接起来，并用规范的方式操作全过程，如同工厂的生产线。常见的软件过程模型有线性模型、渐增式模型、螺旋模型、快速原型模型、形式化描述模型等。这些方法对消除软件开发过程中出现的软件危机起到了一定的积极作用，但是，这些方法和过程也存在着如规则复杂、实施规模不断膨胀和过程繁琐等方面的问题，不能很好地适应目前 Web 应用系统开发的需要。

好的 Web 应用开发方法应该具备以下的主要特征或功能。

(1) 易于掌握。Web 应用开发方法应该能对 Web 应用的各个方面进行全面详细的描述。为平衡表达能力和掌握难度，可采用的方法是封装细节，对那些具有共性的细节进行抽象，以模式的方式提供选择，为开发人员提供高层次的设计概念和方法，掩藏了细节，在不牺牲表达能力的基础上降低了掌握难度，但必要时设计人员仍然可以对细节进行设计以满足特殊的情况。另外，Web 应用设计方法应该充分考虑用户已有的设计经验和技术。

(2) 对复杂系统建模的能力。Web 应用涵盖从简单的静态 Web 应用到动态交互的 Web 应用。由于 Web 应用发展迅速，特别地，当大量传统信息和数据库系统被移植到 Web 环境下，一种新型的 Web 应用出现了，这些程序利用 Web 平台支持和执行商业过程以及工作流，例如出租和预订服务、虚拟拍卖、在线保险等。成熟的 Web 应用开发方法应该能适应这种需求。

(3) 展示层建模的能力。传统的 Web 应用设计方法一般不是很重视用户界面设计，而和传统的应用软件相比较，Web 应用的很多高级功能体现在展示层，这需要提高界面的设计质量。同时，Web 应用设计方法一般不仅给设计人员使用，而且需要给美工设计师、编辑等使用，他们更关心系统的展示设计。所以 Web 应用开发方法需要能针对这些特点对展示

层建模。

（4）系统定制的支持。成功的Web应用应该具有丰富的功能、易于使用的界面和良好的导航结构等特性。而为了达到更高的用户满意度，一种主要的技术是通过个性化定制把合适的内容在合适的时间分发给合适的人。开发方法需要提供系统定制能力，这主要通过对用户的定义和描述来完成，其中包括对用户分组以及用户之间关系的处理。

（5）模型集成和连通的能力。Web应用能够在较高的抽象层次上表达系统和资源是如何集成的。一方面，在很多组织中，新开发的Web应用需要和已存在的业务系统密切相关。这些业务系统可能基于不同的平台，采用不同的实现语言开发而成。开发方法应该能支持与这些遗留系统的无缝连接；另一方面，组件的集成大部分依赖于接口的描述，开发方法应提供精确的和无二义性的对组件接口建模和文档化的能力。最后，Web应用需要和大量的资源和信息服务等相连接，这些可能不局限于组织内部，开发方法应提供表达和存取机制。

（6）工具和文档化支持。理想的工具应能支持在用户参与下，完成从需求确定到实现维护的整个开发过程。丰富的文档支持是设计人员能否掌握开发方法的重要方面。可以说，工具和文档的支持能力是开发方法能否得到广泛应用的关键。

RUP吸收了多种开发模型的优点，具有很好的可操作性和实用性。RUP又是可裁剪的软件开发过程框架，各组织可以根据自身及项目特点对RUP进行裁减，定制出适合自己的过程模型。除了RUP之外，敏捷开发方法也使得“让软件开发更加简单而有效”成为可能。

2.2.1　RUP

RUP（Rational Unified Process，统一软件开发过程）是一套软件工程方法，主要包含：用于成功开发软件的一组核心概念和做法，过程模型和相关联内容库，以及底层过程定义语言。

1. RUP核心概念

在RUP中，软件开发生命周期根据时间和RUP的核心工作流划分为二维空间：横轴表示项目的时间维，是过程展开的生命周期特征，体现开发过程的动态结构，用来描述它的术语主要包括周期（Period）、阶段（Phase）、迭代（Iteration）和里程碑（Milestone）；纵轴以内容来组织，为自然的逻辑活动，体现开发过程的静态结构，用来描述它的术语包括活动（Activity）、制品（Artifact）、工作者（Worker）和工作流（Workflow）。

2. RUP里程碑

RUP中的软件生命周期在时间上被分解为4个顺序的阶段，分别是：初始阶段（Inception）、精化阶段（Elaboration）、构造阶段（Construction）和交付阶段（Transition），如图2.1所示。每个阶段结束于一个主要的里程碑（Major Milestone），每个阶段本质上是两个里程碑之间的时间跨度。在每个阶段的结尾执行一次评估以确定这个阶段的目标是否已经达成。如果评估结果令人满意的话，可以允许项目进入下一个阶段。

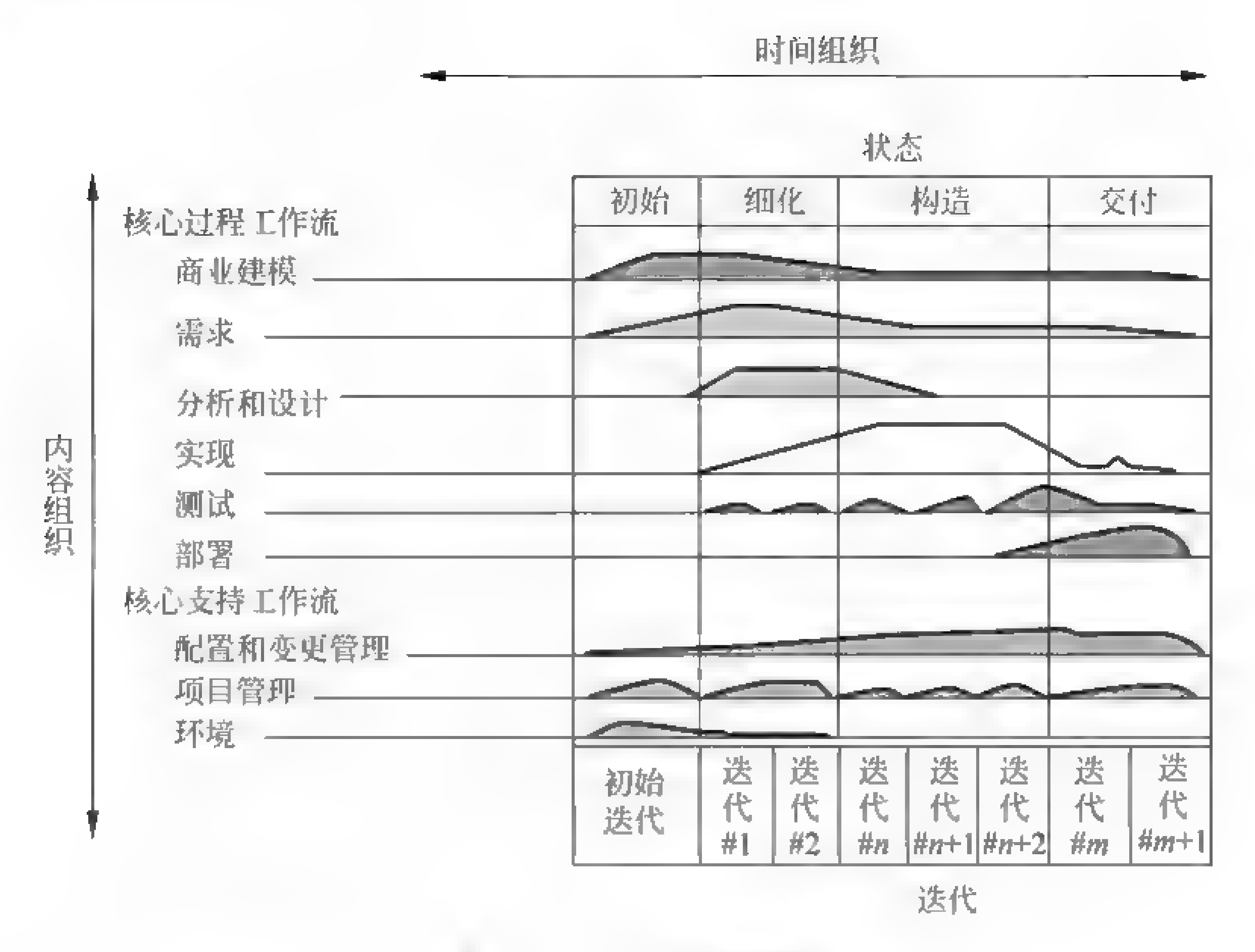


图 2.1 RUP 生命周期

1）初始阶段

初始阶段的目标是为系统建立商业案例并确定项目的边界。为了达到该目的必须识别所有与系统交互的外部实体，在较高层次上定义交互的特性。本阶段具有非常重要的意义，在这个阶段中所关注的是整个项目进行中的业务和需求方面的主要风险。对于建立在原有系统基础上的开发项目来讲，初始阶段可能很短。初始阶段结束时是第一个重要的里程碑：生命周期目标（Lifecycle Objective）里程碑。生命周期目标里程碑评价项目基本的生存能力。

初始阶段的主要目标如下。

① 建立项目的软件规模和边界条件，包括运作前景、验收标准以及希望软件中包括和不包括的内容。

② 识别系统的关键用例。

③ 评估整个项目的总体成本和进度。

④ 评估潜在风险。

⑤ 准备项目的支持环境。

2）细化阶段

细化阶段的目标是分析问题领域，建立健全的体系结构基础，编制项目计划，淘汰项目中最高风险的元素。为了达到该目的，必须在理解整个系统的基础上，对体系结构做出决策，包括其范围、主要功能和性能等非功能需求。同时为项目建立支持环境，包括创建开发案例，创建模板、准则并准备工具。细化阶段结束时是第二个重要的里程碑：生命周期架构（Lifecycle Architecture）里程碑。生命周期架构里程碑为系统的结构建立了管理基准并使项目小组能够在构建阶段中进行衡量。此刻，要检验详细的系统目标和范围、架构的选择以及主要风险的解决方案。

细化阶段的主要目标如下。

① 确保架构、需求和计划足够稳定，充分减少风险，从而能够有预见性地确定完成开发所需的成本和进度。

② 处理在构架方面具有重要意义的所有项目风险。

③ 建立一个已确定基线的架构，它是通过处理架构方面重要的场景得到的，这些场景可以显示项目的最大技术风险。

④ 制作产品质量构件的演进式原型，以减少特定风险。

⑤ 证明已建立基线的构架将在适当时间以合理的成本支持系统需求。

⑥ 建立支持环境(如创建开发案例、创建模板和指南、安装工具等)。

3）构造阶段

在构造阶段，所有剩余的构件和应用程序功能被开发并集成为产品，所有的功能被详细测试。从某种意义上说，构建阶段是一个制造过程，其重点放在管理资源及控制运作以优化成本、进度和质量。构建阶段结束时是第三个重要的里程碑：初始功能(Initial Operational)里程碑。初始功能里程碑决定了产品是否可以在测试环境中进行部署。此刻，要确定软件、环境、用户是否可以开始系统的运作。此时的产品版本也常被称为Beta版。

构造阶段的主要目标如下。

① 优化资源，避免不必要的报废和返工，使开发成本降到最低。

② 尽快达到质量要求。

③ 快速完成有用的版本，如Alpha版、Beta版和其他测试发布版。

④ 完成所有功能的分析、开发和测试。

⑤ 迭代式、递增地开发随时可以发布的产品，也就是要继续描述剩余的用例和其他需求，充实设计，完成实施并测试软件。

⑥ 确定准备好软件系统的外部环境。

4）交付阶段

交付阶段的重点是确保软件对最终用户是可用的。交付阶段可以跨越几次迭代，包括为部署和发布做准备的产品测试，基于用户反馈的少量的调整。在生命周期这一点上，用户反馈应主要集中在产品调整、配置、安装和可用性等问题，所有主要的结构问题应该已经在项目生命周期的早期阶段解决了。在交付阶段的终点是第四个里程碑：产品发布(Product Release)里程碑。此时，要确定目标是否实现，是否应该开始另一个开发周期。在一些情况下这个里程碑可能与下一个周期的初始阶段的结束重合。

交付阶段的主要目标如下。

① 进行Beta测试，按用户的期望确认新系统。

② Beta测试和相对于正在替换的遗留系统的并行操作。

③ 转换操作数据库，培训用户和维护人员。

④ 市场营销，进行分发和向销售人员进行新产品介绍。

⑤ 与部署相关的工程：接入、商业包装和生产、销售介绍、现场人员培训。

⑥ 根据产品的完整前景和验收标准，对部署基线进行评估。

3. RUP 核心工作流

RUP 中有 9 个核心工作流，分为 6 个核心过程工作流(Core Process Workflows)和 3 个核心支持工作流(Core Supporting Workflows)。尽管 6 个核心过程工作流可能使人想起传统瀑布模型中的几个阶段，但应注意迭代过程中的阶段是完全不同的，这些工作流在整个生命周期中一次又一次被访问。9 个核心工作流在项目中轮流被使用，在每一次迭代中以不同的重点和强度重复。

1) 商业建模

商业建模(Business Modeling)工作流描述了如何为新的目标组织开发一个构想，并基于这个构想在商业用例模型和商业对象模型中定义组织的过程、角色和责任。了解目标组织(将要在其中部署系统的组织)的结构以及机制，了解目标组织中当前存在的问题并确定改进的可能性，确保客户、最终用户和开发人员就目标组织达成共识，导出支持目标组织所需的系统需求。

2) 需求

需求(Requirements)工作流的目标是描述系统应该做什么，并使开发人员和用户就这一描述达成共识。为了达到该目标，要对需要的功能和约束进行提取、组织、文档化，最重要的是理解系统所解决问题的定义和范围。

3) 分析和设计

分析和设计(Analysis & Design)工作流将需求转化成未来系统的设计，为系统开发一个健壮的架构并调整设计使其与实现环境相匹配，优化其性能。分析设计的结果是一个设计模型和一个可选的分析模型。设计模型是源代码的抽象，由设计类和一些描述组成。设计类被组织成具有良好接口的设计包和设计子系统，而描述则体现了类的对象如何协同工作实现用例的功能。设计活动以架构设计为中心，架构由若干结构视图来表达，结构视图是整个设计的抽象和简化，该视图中省略了一些细节，使重要的特点体现得更加清晰。体系结构不仅是良好设计模型的承载媒介，而且在系统的开发中能提高被创建模型的质量。

4) 实现

实现(Implementation)工作流的目的包括以层次化的子系统形式定义代码的组织结构；以组件的形式(如源文件、二进制文件、可执行文件)实现类和对象；将开发出的组件作为单元进行测试以及集成由单个开发者(或小组)所产生的结果，使其成为可执行的系统。

对照实施子系统的分层结构定义代码结构，以构件(如源文件、二进制文件、可执行文件以及其他文件等)的方式实施类和对象；对已开发的构件按单元来测试，并将各实施人员(或团队)完成的结果集成到可执行系统中。

5) 测试

测试(Test)工作流要验证对象间的交互作用，验证软件中所有组件的正确集成，检验所有的需求是否已被正确地实现，识别并确认缺陷在软件部署之前被提出并处理。RUP 提出了迭代的方法，意味着在整个项目中进行测试，从而尽可能早地发现缺陷，从根本上降低修改缺陷的成本。测试类似于三维模型，分别从可靠性、功能性和系统性能来进行。

核实对象之间的交互，核实软件的所有构件是否正确集成，核实所有需求是否已经正确

实施，确定缺陷并确保在部署软件之前将缺陷解决。

6）部署

部署(Deployment)工作流的目的是成功地生成版本并将软件分发给最终用户。部署工作流描述了那些与确保软件产品对最终用户具有可用性相关的活动，包括软件打包，生成软件本身以外的产品，安装软件，为用户提供帮助。在有些情况下，还可能包括计划和进行Beta测试版，移植现有的软件和数据以及正式验收。

7）配置和变更管理

配置和变更管理(Configuration & Change Management)工作流描绘了如何在多个成员组成的项目中控制大量的项目活动。配置和变更管理工作流提供了准则来管理演化系统中的多个变体，跟踪软件创建过程中的版本。工作流描述了如何管理并行开发、分布式开发，如何自动化创建工程，同时也阐述了对产品修改原因、时间、人员保持审计记录。

8）项目管理

项目管理(Project Management)平衡各种可能产生冲突的目标，管理风险，克服各种约束并成功交付使用户满意的产品。其目标包括为项目的管理提供框架，为计划、人员配备、执行和监控项目提供实用的准则，为管理风险提供框架等。

9）环境

环境(Environment)工作流的目的是向软件开发组织提供软件开发环境，包括过程和工具。环境工作流集中于配置项目过程中所需要的活动，同样也支持开发项目规范的活动，提供了逐步的指导手册并介绍了如何在组织中实现过程。

2.2.2　XP

极限编程(Extreme Programming，XP)源于快速响应问题域频繁变化的需求，是敏捷过程的一种具体形式，提供敏捷方法(Agile Method)最一般的原则的指导方针。在某些方面，XP可以看做是“超级程序员”描述他们对编码世界的想法。类似地方法学包括并列争求法(Scrum)、自适应软件开发(Adaptive Software Development，ASD)和水晶(Crystal)方法。

XP是一个轻量级的、灵活的软件开发方法，同时也是一个非常严谨和周密的方法，强调创建客户满意的系统，强调团队工作。XP从沟通、简单、反馈、尊重和勇气5个方面改善任何一个软件项目。

2000年，美国软件工程专家Kent Beck对XP这一创新软件过程方法论进行了解释：“XP是一种轻量、高效、低风险、柔性、可预测、科学而充满乐趣的软件开发方法。”即，XP是一种近螺旋式的开发方法，它将复杂的开发过程分解为一个个相对比较简单的小周期。通过积极的交流、反馈以及其他一系列的方法，开发人员和客户可以非常清楚开发进度、变化、待解决的问题和潜在的困难等，并根据实际情况及时地调整开发过程。图2.2展示的是一个简单的XP过程。

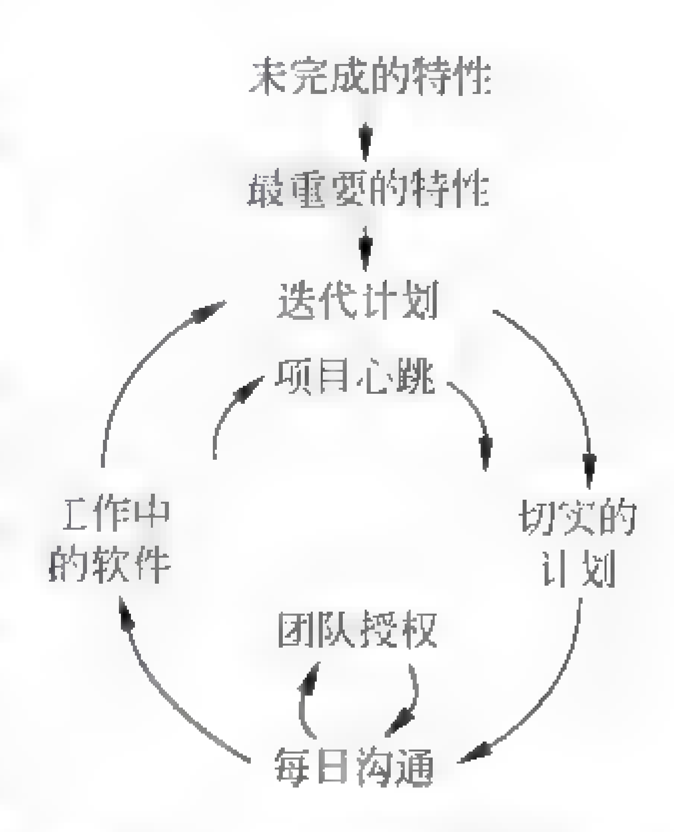


图2.2　一个典型的XP过程

1. 简单规则

XP最令人吃惊的特性是如下一组简单的规则。

1）计划

（1）编写用户故事。用于创建发布版本计划会议的时间评估和替代需求文档，通常由客户编写，与使用情景类似。

（2）制定发布版本计划。勾画出整个项目，将项目拆分成多个迭代，并为每个迭代制定迭代计划。发布版本计划指明哪些用户故事将在哪个系统版本中何时实现。

（3）不断创建小的发布版本。有些开发团队需要在一两天或者至少一二周就要给客户发布版本。每个迭代在结束之前都应该经过测试，展示给客户工作正常性能良好的系统。这对技术人员做出技术决策和业务人员做出业务决策起到举足轻重的作用。

（4）项目分为多个迭代。迭代开发为开发过程添加了敏捷性。以1～3周为限，将项目开发周期拆分为多个迭代。最好选择是1周。将每个迭代长度在整个项目周期中保持为常量。这样就是项目的“心跳”。这个常量使得XP的进度可度量，计划简单而且可靠。

（5）迭代计划。计划是持续的、循序渐进的。每个迭代开始之前进行编程任务计划，每个迭代的长度为1～3周。每个迭代周期结束时，开发人员就为下个周期估算候选特性的成本，而客户则根据成本和商务价值来选择要实现的特性。

2）管理

（1）营造开放的工作场所。沟通对XP开发团队而言尤为重要，因此，XP项目为所有参与者营造一起工作、使人感觉平等无拘束的一个开放的工作场所。这个场所中，提供了开发者之间无屏障的大的每天站立会议场所，提供了会议桌，墙壁上随意悬挂着大幅的、显著的图表以及显示项目进度等。

（2）设置可持续的速度。项目团队只有持久才有获胜的希望。他们以能够长期维持的速度努力工作，他们保存精力，他们把项目看做是马拉松长跑，而不是全速短跑。认真对待迭代的结束时间，每个迭代都是完整、测试充分、集成以及可供使用的。如果无法达到这个要求，重新召开迭代计划会议，重新确定迭代范围。

（3）每天第一件事是举行简短的站立会议。其目的是全队沟通，简要介绍昨天完成的工作、今天计划完成的工作以及哪些问题造成延期。即使没有发言，也可以获得项目的问题、解决方案以及促进团队注意力。

（4）度量项目速度。根据用户故事在迭代中完成的数量进行度量项目中已经完成的工作量，在迭代中统计完成的任务。

（5）让开发人员动起来。使开发人员在开发中切换工作，使其得到交叉训练，以避免在某个领域只有一个人能够完成工作的情况。

（6）XP无效时进行修复。开发过程出了问题就针对项目需求进行修复。

3）设计

（1）保持简单而简洁的设计。简单设计容易完成，保持设计恰好和当前的系统功能相匹配。如果遇到复杂的问题而难以设计，替换为简单的来完成。这些设计通过了所有的测试，不包含任何重复，表达出了设计者想表达的所有东西，并且包含尽可能少的代码。

（2）选择系统隐喻。系统隐喻本身是具有一定质量的简单设计，最重要的质量是能够

无需大量文档的情况下将设计介绍给新的人员。将整个系统联系在一起的全局视图，它是系统的未来影像，是它使得所有单独模块的位置和外观变得明显直观。如果模块的外观与整个隐喻不符，那么你就知道该模块是错误的。

(3) 使用 CRC 卡片进行设计。CRC 卡片的最大价值在于其面向对象的思想。

(4) 创建微小系统以降低风险。创建微小的解决方案以攻克难度高的技术或设计问题，这个方案也是一个非常简单的潜在的解决方案。

(5) 尽早添加功能。只要认为后期需要，就添加额外的一些功能。

(6) 一有可能就进行重构。由于不断地使用或者重用代码，使得系统可维护性变差。XP 提倡剔除冗余，删除无用的功能，对过时的设计重新进行设计，保持代码尽可能地干净、具有表达力。重构遍布于项目的整个生命周期中，以节约时间并提高质量。

4) 编码

(1) 客户始终在场。客户作为开发团队的成员，并在 XP 项目的所有阶段提供面对面的沟通。

(2) 编码要遵循标准。编码标准能保持代码一致性并提供可读性和可重构性，系统中所有的代码看起来就好像是单独一人编写的。

(3) 代码要先进行单元测试。先编写测试代码，再编码，有助于快速而简单地进行编码。

(4) 所有的编码工作都结对完成。所有的产品软件都是由两个程序员并排坐在一起在同一台机器上构建的，结对编程能提高软件质量而不影响交付时间。

(5) 每次只有一对进行集成。为了解决并行开发所造成的集成问题，进行顺序集成。

(6) 持续集成。只要可能，开发人员应该每隔几个小时就集成和提交一次代码，任何情况下都不应该超过 1 天。

(7) 设置专用的集成计算机，进行控制版本。

(8) 代码集体拥有权。鼓励每个人为项目做出贡献，人人都可以修改任何功能任意行代码，添加功能，修订缺陷，改进设计或重构，即每个人都可以参与任何其他方面的开发。

5) 测试

(1) 所有代码必须进行单元测试。先编写测试代码，再编写代码。

(2) 所有代码必须在发布之前完全通过单元测试。

(3) 当发现问题时创建测试。创建接受测试，以防其再次出现。

(4) 经常运行接受测试并公布得分。基于用户故事来创建接受测试，在迭代计划时，用户故事就会被转化为接受测试。客户可以根据脚本语言来定义出自动验收测试来表明该特性可以工作。

所有上述这些规则看似别扭，甚至不成熟，但是它们都是建立在坚实的价值和原则基础之上的。这些原则指明了团队成员之间的期望，但并非最终目标。这些规则定义了提升团队协作和权力的环境，那才是目标。XP 上述每一条规则本身并没多大意义，很像一个由很多小块拼起来的智力拼图，单独查看每一小块都没有什么意义，但拼装起来后，就是一幅完整而美丽的图画，如图 2.3 所示。客户乐意作为开发过程的一员，开发者不论经验都是贡献者，管理人员关注沟通和关系，无用工尽量降到最低以减少开支和对其他人的打击。

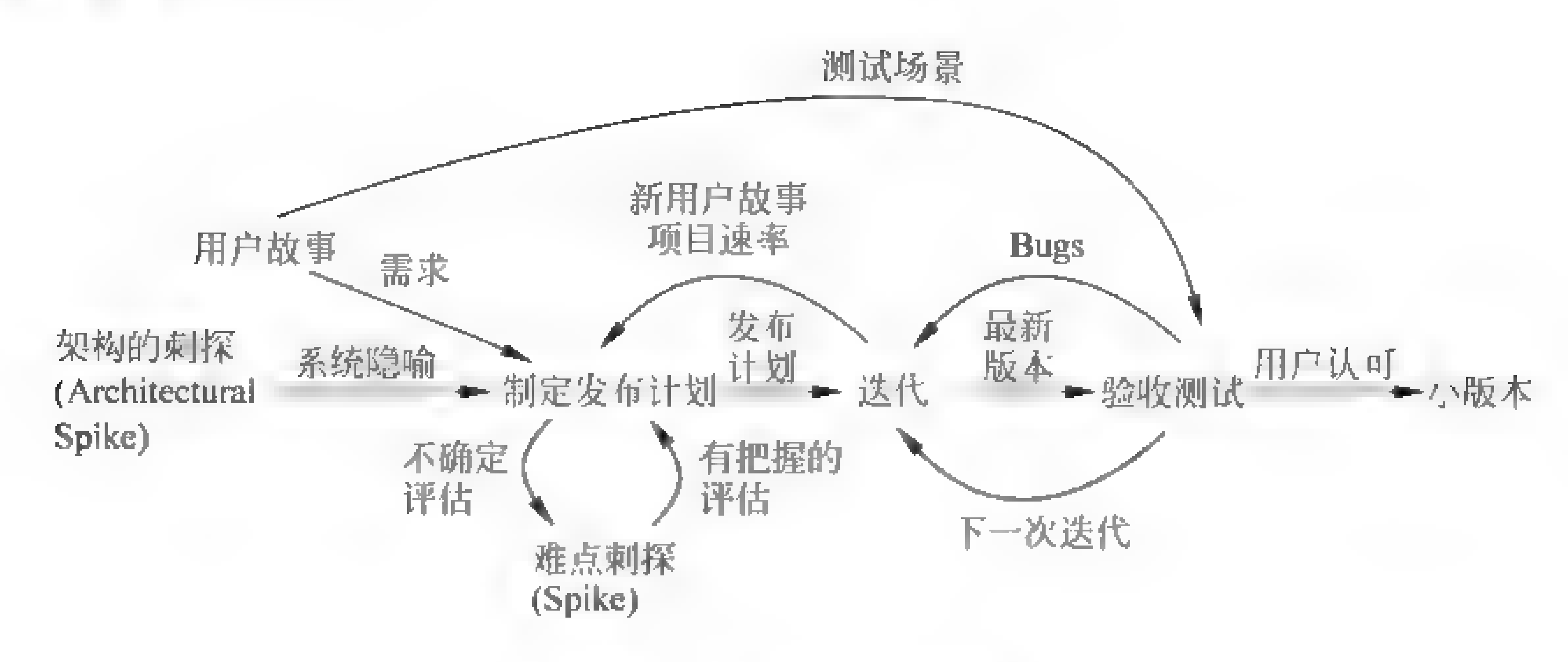


图 2.3 XP 项目流程图

2. XP 的核心价值

XP 的 5 个核心价值是：沟通(Communication)、简单(Simplicity)、反馈(Feedback)、尊重(Respect)和勇气(Courage)。

(1) 沟通。项目中的所有问题都可以追溯到某些关键点上，都是因为相关人员没有能够就一些重要问题进行充分、及时的交流而造成的。XP 认为沟通是必不可少的，开发人员要不断地与客户和其他开发人员进行沟通。

(2) 简单。XP 建议在考虑过程和编写代码时，永远都要使用尽可能简单的方式工作，保持简单而简洁的设计。按照 Kent 的说法，"XP 就是预测。它通过今天做最简单的事情……来避免将来做更复杂但可能永远也不会用到的事情"。

(3) 反馈。通过从第一天就开始测试系统来获得反馈。及时经常地从客户、团队以及真正的最终用户那里听取具体的反馈意见，会让开发者有更多的机会尽可能早地改进其工作质量。这样就可以把握住正确的方向，从而少走弯路。

(4) 尊重。每个团队成员相互尊重并且自重。每个团队成员的任何小的成功都具有其独特的贡献，哪怕只是热情，都应该获得应有的尊重。开发人员应该和客户相互尊重，管理者应该尊重每个成员的工作职责和权威性。

(5) 勇气。尽早地交付系统并根据建议进行改进。如果没有以最快的速度前进，就会失败。在遇到困难时勇气会给开发者力量以克服它，例如，当必须放弃某些代码或者事后做一些变更的时候有勇气这么做。

XP 用"沟通、简单、反馈、尊重和勇气"来减轻开发压力和包袱，无论是术语命名、专著叙述内容和方式、过程要求，都可以从中感受到轻松愉快和主动奋发的态度和气氛。这是一种帮助理解和更容易激发人的潜力的手段。

XP 程序员能够勇于面对需求和技术上的变化。XP 精神可以启发开发人员如何学习和对待快速变化、多样的开发技术。成功学习 XP 的关键，是用"沟通、简单、反馈、尊重和勇气"的态度来对待 XP；轻松愉快地来感受 XP 的实践思想；自己认真实践后，通过对真实反馈的分析，来决定 XP 对自己的价值；尊重所有人的贡献；有勇气接受它，或改进它。

XP 在开发中必须注意这 5 个价值，从而，开发人员能够勇敢地应对不断变化的需求和技术。

3. XP的三个重点

1）角色定位

XP把客户非常明确地加入到开发团队中，并参与日常开发与沟通会议。客户是软件的最终使用者，使用是否满意一定以客户的意见为准。不仅让客户参与设计讨论，而且让客户负责编写用户故事（User Story），也就是功能需求，包括软件要实现的功能以及完成功能的业务操作过程。用户在软件开发过程中的责任被提到与开发人员同样的重要程度。

2）敏捷开发

敏捷开发追求合作与响应变化。迭代就是缩短版本的发布周期，缩短到周、日，完成一个小的功能模块，可以快速测试并及时展现给客户，以便及时反馈。小版本加快了客户沟通反馈的频率，功能简单，在设计、文档环节大大简化。在XP中，文档不再重要的原因是因为每个版本功能简单，不需要复杂的设计过程。XP追求设计简单，实现客户要求即可，无须为扩展考虑太多，因为客户的新需求随时可以添加。

3）追求价值

XP把软件开发变成自我管理的挑战，追求沟通、简单、反馈、尊重、勇气，体现开发团队的人员价值，激发参与人员的情绪，最大限度地调动开发者的积极性。情绪高涨，认真投入，开发的软件质量就大大提高。结对编程就是激发队员才智的一种方式。

XP把软件开发过程重新定义为聆听、测试、编码、设计的迭代循环过程，确立了测试→编码→重构（设计）的软件开发管理思路。

XP的一个成功因素是重视客户的反馈——开发的目的就是为了满足客户的需要。XP方法使开发人员始终都能自信地面对客户需求的变化。XP强调团队合作，经理、客户和开发人员都是开发团队中的一员。团队通过相互之间的充分交流和合作，使用XP这种简单而有效的方式，努力开发出高质量的软件。XP的设计简单而高效。程序员们通过测试获得客户反馈，并根据变化修改代码和设计，他们总是争取尽可能早地将软件交付给客户。

2.2.3 RUP与XP对Web应用的适应性

Web应用本身是一种软件产品，因此软件工程的方法和原则对Web应用的开发仍然具有实际意义。同时，Web工程的提出与软件工程的产生有相似背景，因此软件工程的研究方法、原理在Web应用开发中可以被借鉴。但是由于Web工程的分布性、导航性、交互性等特性，使得Web应用开发又不同于传统的软件开发，传统的软件工程的方法和原则在Web开发中需要经过调整以适应其自身的特点。

1. RUP对Web应用的适应性

可以根据Web应用的特点，在Web应用开发过程中应用RUP的6个最佳实践理论。

1）迭代开发

迭代开发基于不断地发现、发明和实现，它强调在开发周期的早期辨明项目的风险，以便能够一致地、及时地、富有效率地管理和克服这些风险。这种可控的迭代开发过程可以使Web应用的发布周期更短、更快。

2）管理需求

管理需求是一种系统的方法，用于提取、组织、沟通和管理软件敏感部分或应用的不断变化的需求。由于Web应用的需求经常随市场而变更，因此跟踪市场的发展并在项目开发周期中跟踪这些变化是非常有必要的。

3）基于组件的架构

架构从组件、组件集成方式和组件间交互作用的机制和模式等方面描述了应用软件的结构。鉴于Web应用必须是开放的和可扩展的，并且在当前基础上可以变更，因此使用组件架构是至关重要的。组件架构易于扩充，并能适应正在进行的变化，可以最大限度地重用以前开发的组件和第三方开发的组件。只要可能并合适，就可以购买架构。某些功能如人性化、内容管理、负载平衡以及安全等，都已包含在产品包中。使用包含组件的产品对于公司尽快发布Web应用的解决方案是很重要的。

4）可视化建模

模型可以帮助理解和澄清问题以及解决方案。模型是对实体的简化，用于理解复杂的系统。Web架构是多层的和分布式的，在设计、开发和配置时比较复杂。管理这些复杂的需求需要建立慎重考虑的和清晰的架构和设计。可视化建模符号如基于UML的Web建模提供了表示系统架构和设计的机制。

5）检验质量

质量关注过程和产品。无论是中间过程还是最终产品，都应有高质量。Web产品主要面向公众，失败的代价会很高。在性能和稳定性不好的情况下，公众就会不再关注这个Web应用站点。Web应用的发布周期确定后，尽早地、经常地、自动化地对其进行测试是很重要的。

6）控制变更

Web应用包含许多对象和组件，由许多人创建和编辑，经常并行被开发出来。在这个持续开发的工作和环境中，控制变更是必需的。多个版本和配置的并发管理需要在整个开发周期内进行严格的配置和变更管理。

2. XP对Web应用的适应性

和RUP相比，XP已经建立了Web应用开发公认的方法，以满足Web应用的开发需求。

1）处理短开发周期

快速连续的发布是XP项目的特性之一。迭代也允许构造短开发周期。因为进程是轻量级的，XP可以很好地满足这点需求。因此，可以说XP和其他灵活的进程模型完全符合这种需求。

2）处理需求变更

处理需求变更代表了XP的5个核心价值之一，这也意味着任何长期的计划在XP项目中只能是非常粗糙和初步的。尽量在今天为明天的需要做准备，这种有预见的方法被拒绝。相反，客户的紧密整合和一个快速交换的结果，允许不断地适应需求。这意味着XP和其他灵活的过程模型也完全符合这种需求。

3）固定期限和灵活内容的发布

一个成功的验收测试要先于一个版本的发布。然而，在XP项目中没有什么可以阻止

从一个版本到另一个版本内容的改变。完整的发布计划是灵活的，能接受新的或改变的需求，不断地改变需求和对以后版本的相关验收测试，这意味着 XP 也满足这种需求。

4）不同版本的并行开发

XP 并没从根本上排除不同版本的并行开发，然而，关键问题是需求计划，因为 XP 项目很难使用有预见的工作方法。另一方面，Web 应用团队之间的计划和协议对后续版本目标和内容的交流是必要的。事实上，如果工作于不同版本的 Web 应用团队成员之间允许私下沟通，例如，如果交流的第一个 XP 核心价值被应用，那么这种沟通通常在 XP 项目中是行之有效的。为了使整个项目状态和过程透明，关于版本的短报告形式的会议将定期举行。关于内容的部分可以在会议外讨论，例如，开发人员负责解释会议上提出的问题。原则上讲，灵活的过程模型能满足这种需求。

5）重用和集成

由于在 Web 应用开发中巨大的时间压力，重用已有组件非常重要。对已有组件的集成需要方法论的支持而不是过程本身的支持。另一方面，通常一个开发过程创建的新组件在别的开发过程中也可以重用。基本上，在 Web 应用开发过程中，当别的开发过程正在并行运行时，可重用组件被开发，这种方法是行之有效的。在这种情况下，提供可重用组件的开发过程将被看做在 Web 应用中有需求的外在客户。这意味着在 XP 过程中对应用客户在现场的原则是有益的。这允许在两个过程中开发的组件相互协调、相互反馈。然而，应该注意因为 XP 过程对特定问题的解决进行了优化，所以这种方法可能不易实现。在这样的情况下，使用专门为重用软件设计的过程要比 XP 或敏捷过程模型更加合适。

6）适应 Web 应用的复杂性水平

Web 应用开发的早期关于 XP 的优点是能处理需求不清楚的情况，使得较短的开发周期之后 Web 应用就能被使用。然而，如果应用逻辑和内容的复杂性增加，那么 XP 会成为一个不太合适的过程。在这种情况下，过程最好支持大的开发团队来进行分布式开发，而 XP 或是敏捷过程模型通常是不合适的。

2.3 定制基于 RUP 和 XP 的 Web 应用开发过程

开发 Web 应用与开发其他应用软件有许多相似点，但也有许多值得注意的不同之处。Web 应用的外观和界面是普通用户关注的焦点，同时也必须为行业的专业人员所接受，如市场人员、创意设计师和商务执行官。因此，对于 Web 开发人员来说，挑战不仅仅在于要掌握不断更新的开发技术，还必须使用一种能促进所有利益相关者达成一致意见的开发过程。

2.3.1 基于 RUP 和 XP 的 Web 应用开发过程

基于 RUP 和 XP，描述一种适用于 Web 应用的过程。该 Web 应用过程既吸取了 RUP 过程中设计与文档的特点，又遵守了 XP 的快速开发、重构、测试先行等原则；既适合了现代软件开发的实际情况，又不必使开发人员陷入过分设计而导致开发进度缓慢的困境，最终达到快速、和谐地开发系统的目的。

1. 迭代开发

Web应用开发过程是一个迭代的过程，如图2.4所示。迭代开发是RUP和XP都具有的特点。对Web应用开发过程而言，在捕获需求阶段应尽可能获取系统的所有需求，经过设计、实现形成了迭代1的成果，可以让用户评估这些结果是否满足了用户的要求。在迭代1的过程中，可以提出新的需求，如由用户提出的或在开发过程中由开发人员提出的，在下一次迭代中捕获这些需求并建立为用例，在几次迭代后就形成了最终系统。

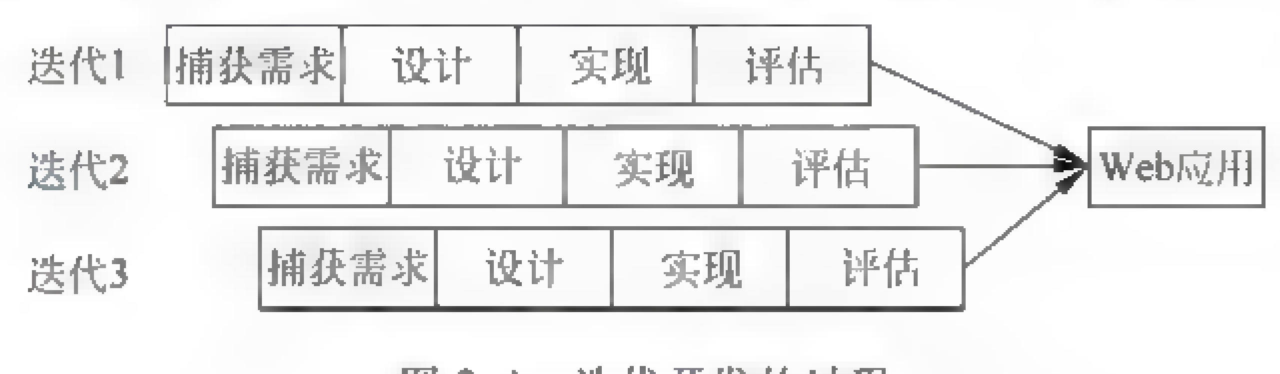


图2.4 迭代开发的过程

1）第一次迭代

需求工作的主要任务是：分析问题，了解利益相关者的需要，创建用户故事以及定义系统的用例模型。设计工作的主要任务是：生成导航图，显示站点用户将如何浏览站点。实现工作的主要任务是生成"创意设计构件"，构建关键Web页面设计的实体模型表示。与用户沟通并评估导航图、创意设计构件和Web页面的实体模型。

2）第二次迭代

需求工作的主要任务是：根据第一次迭代的用户评估细化创意设计构件和Web页面的实体模型，选择系统中最重要的和最具架构意义的用例作为第一个增量的需求，并为Web应用定义相对良好的架构，包括一组定义良好的架构机制（如Web浏览器、Java Applet、Servlet、ASP和JSP等）。设计工作的主要任务是：建立功能性用户界面原型，初始Web UI原型通常只支持系统中最重要的和最具架构意义的用例，并确定第一个增量所对应的用例；分析用例并确定和优化执行用例事件流的分析类，使用UWE等Web建模方法对Web应用架构建模。实现工作的主要任务是：创建Web设计元素，汇集建立Web应用的Web页面的分散图形图像，包括Web页面、Applet、脚本、图形和在浏览器环境中执行的其他元素。通过测试用户交互评估来验证Web应用的结构适合其用户，并将开发的增量发布在Web环境中。

3）第三次迭代

根据第二次迭代中的需求工作，确定第二个增量的用例，并继续分析用户故事和用例，确定和创建Web设计元素，实现第二个增量。最后经过用户交互测试，并以增量和连续的方式发布在Web环境中。

2. Web应用需求捕获

Web应用的需求工作的主要任务是：Web应用需求捕获和用户故事以及基于用例的Web需求描述。构建Web应用与开发传统软件相比，要涉及到更多的人员。这些人员通常包括商业执行官、市场人员、创意设计人员、客户支持人员和技术研发团队等。拥有一种能够促进这些人员相互沟通的过程是成功的关键。

用例提供了项目相关人员如用户、经理、艺术指导、架构师和程序员等相互进行联系的通用语言,任何人都可以使用这种语言与项目取得沟通,并描述Web应用将做些什么。它允许人们用商业解决方案方面的语言进行交谈,每个人都根据用例确定的规则来表达。

成功的Web应用起始于引人注目的视觉外观。这种视觉外观必须由该Web应用的利益相关者来亲自开发,并且要确保一开始就和项目的目标保持一致。视觉表现应符合以下目标。

① 必须与要解决的问题相适应。

② 能明确定义系统的边界。

③ 描述出系统最重要的特征。

一旦视觉外观取得一致意见,项目的利益相关者需要进行进一步的沟通,以界定系统的外围环境,以及描述系统应该为这些用户提供的交互服务和系统行为。具体步骤如下。

首先,分析人员与系统的用户进行交流,捕获系统潜在的使用者。系统分析人员界定系统的外围环境,谁将是系统的最终使用者,这些使用者可能是人,也可能是一个外部系统,并将这些使用者分类成各种角色。

其次,捕获不同角色与系统交互的过程。系统分析人员与每一种角色中的一个人员进行交流,捕获他与系统交互的过程。这些过程不但包括成功执行的过程,还包括在发生意外情况时,如何执行工作流。在捕获系统的一个需求后,就应当为其建立文档。由于文档要让用户来评估,因此,文档的书写形式应采用客户语言书写,使客户容易理解。

最后,将使用客户语言书写的用例文档建立为用例图和活动图。同时,在规格书中加入非功能性需求的描述,一些通用的术语也应该加入到词汇表中。增补规格书包含了一些对通常需求(对整个系统而言)的描述,如可用性、稳定性、性能、安全性、Web集群和可支持性等。这个词汇表保证了项目成员对重要的概念保持统一的理解。

Web应用把许多不同类型的利益相关者组织起来,涉及到不同的学科,包括市场、技术以及许多内部和外部的组织单元。例如,一个B2B的Extranet解决方案通常涉及到组织内部的不同利益相关者,以及可能成为该Web应用的客户组织的内部利益相关者。一个消费者站点将涉及到客户服务的其他组织部门的利益相关者,以及真正的顾客。为了定位这些需求,使用交流促进会来沟通是必需的。在RUP中,可以采用需求工作室(Requirements Workshop)、用例工作室(Use-Case Workshop)和用例分析工作室(Use-Case Analysis Workshop)等的指南。

3. Web应用设计

用户界面设计的好坏对Web应用的设计和成败都是十分关键。由于在Web应用的市场环境中,客户并不是购买软件并安装在他们的机器上,而是在Web上冲浪,并且立即判断出Web页面是否吸引他。要构建成功的Web应用,关键之一,就是要使Web应用程序有引人注目的外观。

1) 构建创意设计大纲

开发Web应用需要更加重视Web页面的创意设计。在定义了参与者和用例的同时,就已经得到了初步的用户界面方案。在Web应用设计阶段,应该更进一步地细化用户界面方案,得出创意设计大纲。例如网站设计,创意设计大纲定义了如下内容。

① 网站的总体风格(如这个网站是权威性的,或是轻松幽默的,或是服务性的?是偏重于保守的还是充满煽动性的?)。

② 用户将以什么样的方式来访问这个网站(如他们的连接速度如何?)。

③ 用户将使用什么样的浏览器。

④ 网站是否使用了框架结构。

⑤ 网站在色彩的使用上是否有限制。

⑥ 如果有用,需要定义一个色彩标准指南(包含网站的 Logos 和总体色调的标准)。

⑦ 需要一些什么样的动态效果点缀(如鼠标翻滚效果、动画、滚动新闻、多媒体等)。

2) 设计导航图

导航图是 Web 应用中的一种用层次树方式显示的视图,能够描述网站的用户如何访问该网站。导航图的每一层显示出到达该层对应的屏幕页面需要多少次点击。通常,总是希望只需在起始页(通常被称做主页)上点击一次,即可到达你的 Web 应用站点上最重要的区域。

在项目的早期确定网站导航图,可以提供一种有效的沟通媒介,便于项目的利益相关者和项目开发团队之间交流。用户也可以更好地想象浏览该 Web 应用的情况,创意设计人员可以更好地了解网站导航模式。用例模型用于描述系统向最终用户提供怎样的服务,因此导航图是自然地从用例模型进化而来的。

导航图是"用例故事板"技术的变种,"用例故事板"在 RUP 的"用户接口建模"行为中进行了定义。要开发导航图,首先应为每一个用例定义一个主窗口或主页面。在这个阶段,我们还不能确切地知道每一页看起来会是什么样子,甚至不能确切知道到底会有哪些具体的页面。所以人们关注于定义"逻辑页面"。这些逻辑页面是可选的,将随着用户接口的发展完善被采纳或被抛弃。逻辑页面用 UML 的构件 — 边界类来分析描述。

随后在设计和实现阶段,我们将用 HTML 页面或其他可视元素来替代 UML 的描述。一旦逻辑页面被定义了,导航图着眼于描述用户如何从一个逻辑页面浏览到另一个页面,同时描述了逻辑页面提供的主要特性。对大型系统而言,人们通常会为每一个活动者定义一个导航图。为了更深入地挖掘细节,每一个用例也需要定义同类的视图,以便定义那些用户在执行用例时将会浏览到的更多的页面(边界类)。当这些页面(边界类)被定义时,同时也描述了它们所处理的信息内容。

3) 设计用户创意设计方案和界面原型

界面方案向利益相关者提供了 Web 应用的一些可选的创意设计方案,以便推进界面设计过程,最终得到一个确实吸引人的视觉设计。创意设计方案包含了一些网站的视觉外观模型。这些模型通常是一些有代表性的"平面"图形,由许多帧描绘浏览窗口外观的浏览视图组成。做创意设计方案选择的意图在于先就网站的视觉规范达成一致意见,然后才进行 HTML 界面原型的开发。

RUP 提供了一个活动,"用户界面原型建模",它提供了一个收集用户界面反馈意见的通用方法。在此活动基础上,选择一个最重要的用例,然后开发出许多可选的界面方案(例如至少 10 个以上)。从这些方案集中选出三个最有希望的设计提交给利益相关者。通常利益相关者就最终的 Web 设计方案达成一致意见以前会有三次迭代。这是一个界面设计和迭代的过程。一旦这个过程达成一致意见并终止后,界面方案将进一步开发成具有实际功

能的用户界面原型。

4）设计 Web 设计元素

Web 设计元素指的是那些被组合到一个网站各个页面中的零散的图形图像。保证网站上用户界面的一致性对于保证网站的可用性来说是必需的。网站应该给用户提供前后一致的浏览体验。为了做到这点，项目开发过程中必须在网站中统一使用一整套标准的图形组件。这些图形组件应该在项目起始之初就设计好，还应设计如何使用这些图形组件的指南，以便项目组的全体成员都明白何时以及如何使用这些组件。

例如，Web 设计元素可以包含像导航图标和页面背景等图形元素。在整个网站中重复使用高质量的标准图形元素可以保证网站的一致性，并缩短网站投用前所需时间，减少开发费用，同时通过使用一套更高质量的图形也可以提高网站的开发质量。

Web 设计元素是与最初的 Web 用户界面原型一起创建的。设计界面方案中可以挑选加工出用于 Web 用户界面原型的 Web 设计元素，并可以从中确定最终的用户接口原型，同时 Web 设计元素也被确定下来。

5）初始 Web 页面原型

界面方案最终发展成用户界面原型。用户界面原型的外观基于最终确定的界面方案，并在 RUP 的“活动：原型化用户界面”阶段被创建，设计用户界面原型时使用了上面定义的 Web 设计元素。初始 Web 页面原型通常只支持了一小部分系统功能，这个原型是基于最重要和最典型的用例来创建的。

初始 Web 页面原型的开发能促进用户和设计者的交流，更好地沟通对网站的外观和给人的感觉，同时网站相关的功能也被开发出来。在投入主要资金用于开发用户界面和网站功能之前，尽早获得用户对网站的看法是很有必要的。

6）Web 页面指南

Web 页面原型完成后，就可以编写用户界面设计的详细指南。该设计风格指南将指定何时以及如何使用 Web 设计元素、色彩配置、字体、层叠样式表，以及确定导航单元将怎样起作用和放置在哪里。Web 页面指南在“活动：开发用户界面指南”中被定义。

7）架构分析

根据从类似系统或在类似问题域中获得的经验定义系统的候选架构，定义系统的架构模式、关键机制和建模约定。Web 应用通常不安排停机时间。架构可能（并通常会）需要在系统运行的同时提供升级，并在主服务器出现故障或者维护或升级服务器时切换到备用服务器。

8）用例分析

确定执行用例事件流的分析类，包括边界类、实体类和控制类。确定分析类的职责、属性和关联，记录体系结构机制的使用情况。Web 页面本身表示为边界类，数据元素表示为实体类，而服务器端行为（如活动服务器页面、Servlet 之类）通过控制对象来表示。

9）确定设计元素

通过“活动：确定设计元素”优化分析类，确定类、子系统和事件等设计模型元素，并将它们映射到 Web 开发框架中的现有机制，在可能的情况下从以前的项目或迭代中复用现有设计元素。

4. Web应用的构建

构建的目的在于将在设计工作流中形成的设计以代码的形式来实现，从而实现客户预期的需求。在Web应用构建中，构建和获取所有Web应用的内容，并将其集成到Web应用的架构之中。选择合适的产生Web页面的工具集，实现每个页面的布局、功能、表单和导航功能，实现所有的计算功能。

在实现代码阶段要大量地采用重构的方法。尽管已经产生了很多的模式，但在不同环境中，模式不可能解决所有的问题，相反，模式需要开发人员充分发挥自己的才智来补充，所以在一个完整的设计中，对一些不能由模式来解决的问题，需自己来设计。但是，一个设计中，可能由于经验的不足或对业务的理解不够充分可能一开始不能产生很好的设计，也许开始的设计对后来添加的元素不能很好适应，就应该采用重构来将现有的模型改变成更优秀的模型。重构是在不改变行为的前提下，对它的结构进行改变。重构不是对原有模型和原有代码的完全抛弃，而是在原来的基础上，对原有模型和代码改变它的结构，使结构优化。

5. Web应用的测试

测试是保证代码健壮的一种手段，任何代码都要经过测试，因此通过测试是编写健壮代码的目标。将设计测试放于实现之前，可以使代码的编写更具有目的性，它由程序员来完成，这里的测试是一种单元测试，它采用的方法是黑盒测试法，因为代码的结构还没有实现，不可能用白盒测试。同时，测试也是进行重构的保证。测试包括测试用例和测试规程，测试用例是测试时使用的测试数据和应得到的结果，测试规程是测试时采用的步骤。在单元测试中，测试规程都是一段测试代码，一般不会很复杂。但在基于EJB分布式对象技术中测试本地Bean比较复杂，因为它不能被客户端程序调用，因此对本地对象的测试，应编写一个无状态会话Bean来对它进行测试。

在Web应用开发与实现中，采用了测试先行的方法，在设计完测试后，由程序员实现对象，然后将构件交由集成人员将其集成到系统中。

Web应用的测试很大程度上注重于性能测试，以确保Web应用程序可支持并发用户数量的激增。还必须测试用户交互来验证Web应用程序的结构适合其用户。同时还需进行浏览器测试，因为浏览器和浏览器版本之间的兼容性经常会限制用户界面中的设计选项。

在Web应用中，实现可分为三大部分：界面类、控制类、实体类。由于界面类是与客户进行交互的类，它的功能性要求比较少，因此，对界面类的测试一般依靠人来感受界面的效果。在一些需要实体数据时，应采用桩代码。桩代码不实现代码，仅提供接口的基本实现，以使测试可以进行下去，并且这些实现也不必符合业务要求。

6. Web应用的部署

部署Web应用就是将构建好的Web应用发布并部署在用户的环境中，并从最终用户那里得到反馈，建立修改的基础。在部署时需要注意以下几个方面。

① Web应用的产品发行往往是递增式和连续的，而较少注重于传统的介质发布。

② Web环境中的用户培训往往集成到Web站点自身的设计中，这样站点的使用就直观了。传统培训和用户手册或文档的创建工作减少了，而更强调流程前端的图形和内容

设计。

③ Web 环境中的生产应用支持必须注重于在不可预测的负载情况下维持高可用性。它可能还需要使系统能在主服务器出现故障时继续运行，并在系统运行的同时允许服务器升级。

④ 研究用户如何使用应用程序。该信息对于了解谁在使用应用程序和如何使用是有价值的。这些观察结果有助于进一步开发发布版，促进用户交互。

部署一个 Web 应用可以非常简单也可以非常复杂，其主要任务是构建部署计划。基于一台服务器和现有网络的简单 Web 应用容易部署，只需建立服务器就可以了，客户端可能已有了合适的浏览器，其部署计划比较简单。

如果 Web 应用要处理安全问题以及负荷问题，就必须要有一个有效的部署计划。另外，许多系统同步于遗留系统，要和原来的系统一起并发运行。由于 Web 应用架构中涉及的错误处理和负载均衡常常包括第三方和现有的组件，这些组件是需要整合的。Web 应用也需要对网络资源和供给做仔细的计划。大多数大型 Web 应用都有多余的 Internet 连接和站点外的备份系统。对这些类型的 Web 应用进行部署，需要仔细地计划和管理。

7. Web 重用与集成

Web 应用开发的巨大时间压力的一个直接后果是开发人员应该尽可能地去重用已有的组件。如果一个项目团队需要开发一个可重用的组件，它可以和其他需要使用这个组件的开发团队共同开发，将已有应用或者其他正在开发的应用集成起来。

在软件集成中，软件架构就是一种集成架构，它是分布式软件应用系统开发者进行软件集成的指导基础。不同的应用领域有不同的架构，目前比较流行的架构有 Java EE、Windows DNA、CORBA、Web Service 等。

遗留系统是指已经交付并能使用的系统。随着 Web 技术的飞速发展，需要构建新的系统，将系统的业务拓展到 Web 环境下，不仅仅局限于局域网内。如何利用遗留系统实现软件重用，降低再开发成本，一直是软件开发追求的目标。目前流行的实现对象 Web 的主要技术有 DCOM 和 CORBA，却由于互操作性以及安全性等多方面的原因，很难在 Internet 上最大限度地发挥作用。传统上将遗留系统业务拓展到开放环境下的方法是在开放环境下对遗留系统进行重构或通过 CORBA、DCOM 技术与 Web 的结合对遗留系统进行集成。

近几年，Web Service 技术迅速兴起并日趋成熟。Web Service 采用一种面向服务的架构，其中定义了一组标准协议，用于接口定义、方法调用、基于 Internet 的构件注册以及各种应用的实现。基于 Web Service 的应用程序也因此具备松散耦合、面向组件和跨技术实现的特点。这使得基于 Web Service 实现对遗留系统的快速改造、集成和重用成为可能。

大型遗留系统往往有丰富的、复杂的结构。它也可被分割成相对独立的子系统。尽可能地将接口进行封装，就可达到松耦合。如果一个软件实体是独立的、自我包含的、粗粒度的和松耦合的，它将成为软件服务的候选对象。考虑到遗留系统自身的特点，实现遗留系统信息集成会受到多方面的限制，在实现技术方面应满足以下基本要求。

① 遗留系统的信息集成应该是遗留系统具体应用的集成，应支持业务正常运转所需信息的正常交互，而不是遗留系统简单的互联及信息交换。

② 集成是动态的，能根据企业经营策略及变化来及时调整集成方式。

③ 应充分考虑对安全稳定运行的要求，保证遗留子系统在集成后仍能保持单独运行时的安全稳定性。

④ 遗留系统的信息集成应该是对原有遗留子系统功能的扩展和延伸，而不是推倒遗留子系统原有的所有功能。

这里主要介绍将遗留系统进化为 Web Service 的再工程方法。

1）评估遗留系统

对遗留系统进行评估，是为了反映遗留系统的当前状态和说明遗留系统属于生命周期的哪个阶段。根据评估结果，决定再工程决策。如果遗留系统满足以下因素，那么将它移植到 SOA 中是合适的。

① 业务逻辑中包含可重用的且可靠的功能。

② 与维护整个系统相比，遗留系统中的可重用构件更易维护。

③ 从需求的角度来说，能够将功能做成一个独立的服务。

④ 客户端是能分解的。

2）解耦遗留系统

由于遗留系统可能已经长时间的进化，使系统具有高耦合。文档不能反映遗留系统的状态。许多情况下，需要将遗留系统分解成低耦合、高内聚的系统。可采用软件聚集技术来捕捉可重用的、独立的、自我包含的、粗粒度和松耦合的遗留代码段。

3）业务规则抽取

尽管得到独立的和松耦合的构件，但在构件中存在很多无关的代码，这时就需要把核心的业务功能代码段抽取出来，这些代码段的组合就是业务规则。业务规则抽取的步骤如下。

① 识别出每个构件中的输入变量。

② 利用程序切片技术得到输入变量有关的代码段。

③ 识别出每个构件功能相关的输出变量。

④ 利用程序切片技术得到输出变量有关的代码段。

⑤ 找出的与输入变量和输出变量都有关的代码段即为所求的业务规则。

4）业务规则确认

对所抽取的业务规则要进行确认，确认的标准如下。

① 真实的表示：从代码中抽取的规则必须要反映软件的状态。

② 一致性：是否与该软件的域模型一致。

5）服务包装和集成

服务包装是将候选的遗留服务变成功能性服务的必要步骤。因为一个面向服务的体系结构鼓励单个服务自我包含。

① 以抽取的业务规则为基础，设计服务接口。使用 WSDL 对服务的相关数据进行描述。如果目标服务是一个 Web Service，那么就用 SOAP 处理器进行集成。

② 创建 Web Service，并进行注册。UDDI 能使公司和应用快速地、容易地和动态地发现和使用因特网上的 Web 服务。

③ 在面向服务体系结构中通过服务组合来改进遗留系统。为了在面向服务的体系结构中建立更强的合作服务，需将遗留代码中的服务与其他服务或基于服务的应用组成起来。最终的合作服务将更有价值。

从 Web 服务的角度来集成并再工程遗留系统，主要包括程序转换、重构、包装和建模等技术。采用以上的 Web 重用与集成方法，可以使新的系统更容易维护，并降低维护费用。

2.3.2 敏捷 Web 应用开发过程

Web 工程过程活动贯穿于整个 Web 应用生命周期，从应用概念的生成到开发、部署，不断地细化和升级维护系统。为了降低开发 Web 应用的复杂性，需要一个描述开发 Web 应用阶段的过程模型。一个过程模型应该帮助开发人员注意 Web 应用的复杂性，降低开发风险，处理变更的可能性，快速部署 Web 应用，维护 Web 应用。当 Web 项目进行时，能为管理提供反馈。而且，Web 应用的开发过程必须是可监督和跟踪的。为此，给出了一个 Web 工程过程模型，如图 2.5 所示。

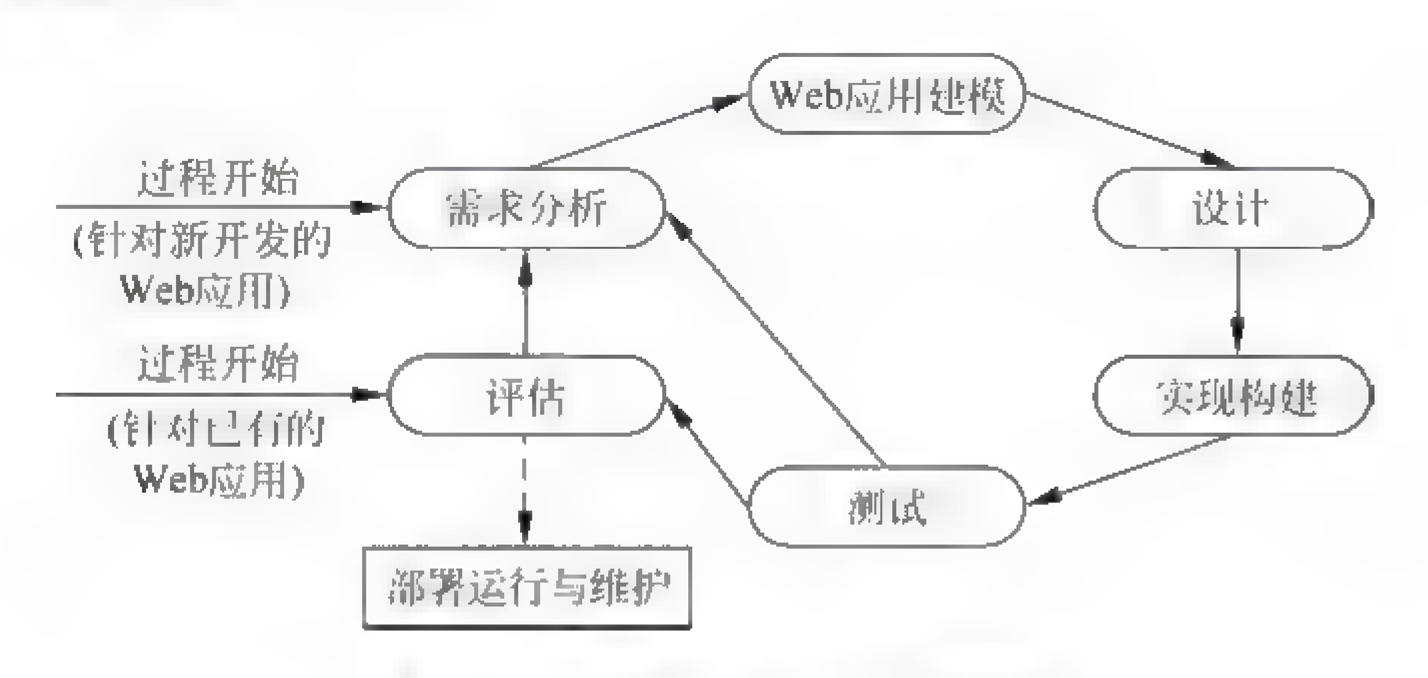


图 2.5 敏捷 Web 工程过程模型

Web 应用开发生命周期包括需求分析、Web 应用建模、Web 应用设计、Web 应用构建、Web 应用测试(包括评估)、Web 应用部署、Web 应用运行与维护。Web 应用开发过程的每个阶段都有可能返回到前面的阶段进行修改，所以 Web 应用的开发过程属于敏捷的、迭代式的开发过程。

(1) Web 应用需求分析与传统软件的需求分析不同，Web 应用需求分析不但要分析 Web 应用本身的功能和性能，还要对可能的用户群体进行分析和调查。(第 3 章)

(2) Web 应用建模包括建模功能、内容、超文本和适应性建模。(第 4 章)

(3) Web 应用设计不但包括 Web 应用架构设计(第 5 章)，还包括交互设计、表示设计、内容设计和功能设计(第 6 章)，如表示设计中包括页面的主色调、页面框架结构、文字颜色搭配、动画和图片设置等。

(4) Web 应用构建包括客户端的构建、服务器端的构建以及相互通信协议，包括后台程序的开发、页面程序的编写和页面的制作。在设计阶段决定的 Web 框架基础上，进行具体页面的设计与制作，把内容提供人员的内容连接到具体的页面。(第 7 章)

(5) 基于 Web 应用的测试包括 Web 应用功能、性能、页面、兼容性、安全性、接口等测试。(第 8 章)

(6) Web 应用测试后，评估如果达到标准，就将 Web 应用发布并部署到 Internet 上，进入运行与维护期。Web 应用需要经常更新，这种更新包括页面内容的更新、页面框架的改变等等。大型 Web 应用的管理是一项艰巨的任务，因为这种更新发生得非常频繁。(第 9 章)

2.4 总结与展望

Web应用能使客户和商业公司具备在线商务处理、供应链集成、动态满足客户要求以及个性化服务等能力，这就要求Web应用的架构是健壮、可伸缩和可扩展的，可以提升性能并且能处理大量的不可预知的商务交易负载。Web应用的开发不同于传统应用软件的开发，有着自身的特性，需要根据Web应用的特性定义一个完整的Web应用开发过程。本章在分析RUP和XP对Web应用的适应性的基础上，给出了具体的Web应用过程模型，用以指导Web应用的开发。

Web应用开发过程是一个迭代的过程。每次迭代中的任务重心也随着整个Web应用开发的推进而变化。由于在Web应用中用户界面受到极大的关注，因而在需求和设计活动中，需要构建创意设计制品和导航图，以细化界面需求并制定界面外观的多个备选设计方案。同时，针对Web应用的多层架构、封装商务逻辑、集成多种异构遗留系统等的特点，介绍了基于Web Service重用和集成遗留系统的再工程方法。

随着Web技术的不断发展和用户需求的不断丰富，很多Web应用也面临可扩展性和复杂性的挑战。因此，在定义Web应用的开发过程时，也必须考虑如何使Web应用能更好地适应高可扩展性和高复杂性的要求。定义整个软件开发项目的元过程(meta-process)，可以帮助管理这些要求。元过程，作为更高层次的过程模型，可以确保项目边界外的战略规划，从整体上保证Web应用的开发符合整体战略规划。元过程主要监控Web应用的特点，标识Web应用的复杂性水平，并根据复杂性水平，实例化为具体的轻量级过程或重量级过程，以及两种过程之间的过渡阶段。例如，元过程的任务之一就是持续监控向更高复杂度过渡的各种迹象，并监管子项目的资源分布和目标。元过程在这里作为一个管理过程以控制变化，而非在构造产品。而且，从长远来看，Web应用的开发也会逐步演化成一个产品生产的过程，这时重量级和迭代过程的优点将更为凸显。

第3章 Web需求工程

需求活动是软件开发过程的重要组成部分，它始于软件项目开发的早期，是整个软件开发过程的入口。需求描述了软件系统的蓝图，说明将要开发的系统需要提供何种功能、达到何种要求等。通常，软件需求包括业务需求、用户需求和功能需求三个层次。

资料表明，软件项目中40%～60%的问题都是需求分析阶段埋下的隐患，而软件开发中70%～80%的返工都是由需求方面的错误导致的。调查显示，Web应用有81%都未达到商业需求。对于Web应用来说，如果处理不好需求，会导致需求获取不完全、不正确、不一致等问题，严重时甚至导致Web应用失败。因此，只有了解、分析、明确Web应用的需求，并且能够准确、清晰地表达给参与项目开发的每个成员，才能保证项目开发过程满足用户需求。

需求工程是为了保证软件系统满足所有利益相关者(Stakeholder)的目标、需要和期望而进行的活动。与传统软件工程一样，Web应用的需求工程的活动一般也可以分为需求获取、需求表示、需求分析、需求确认与验证4部分。传统的软件工程已经总结出一套有效的需求分析的方法与规则，例如用于形式化表达需求的用例图等。Web应用的需求工程可以借鉴传统的软件开发中的方法和手段。但由于Web应用中需求工程面临着特定的问题，在Web工程中还应根据Web应用的特性来实施需求工程。

3.1 Web需求特性

Web应用不同于传统的应用软件，因此Web应用的需求工程也不同于传统的应用软件的需求工程。Web应用需求工程的特性包含如下几方面的内容。

(1) 多学科性。Web应用的开发涉及的学科非常广泛，它需要不同领域的专家的共同参与，如网络安全专家、系统架构师、软件可用性专家、美工设计师、数据库专家、业务领域专家、Web应用开发人员等，这对Web应用的需求的确定提出了巨大的挑战。因此，在定义需求时需要协调各个领域专家对需求的理解。这是Web应用和传统软件应用的一个显著区别。

(2) 利益相关者未知。在获取需求时，Web应用的很多利益相关者(例如未来的实际用户)无法确定，而Web应用开发又需要利益相关者的共同参与才能获取比较完整、实际、可靠的需求，这使得获取需求面临很大的困难。

(3) 不断变化的需求和约束。相比传统的应用软件来说，Web应用的需求和约束常常

处于变更之中，很难稳定下来。在Web应用的开发过程中，随着开发者与用户的不断交流，需求会不断做出修正以接近用户的目标，市场的变化也会迫使Web应用的需求出现变化。

(4) 未知的软、硬件环境。由于用户软、硬件环境多种多样，Web应用的操作环境也是不固定的。对于Web应用开发人员而言，有许多因素需要考虑(例如需要考虑在多种浏览器下Web应用能呈现统一的视觉效果)。

(5) 质量控制。对于Web应用而言，质量的好坏是最关键的。Web应用的质量主要包括性能、可用性和安全性等方面的内容。在构建Web应用之前，Web应用开发人员必须对这些质量属性进行精确分析，但网速慢导致Web应用页面打开缓慢或页面显示不全等问题是开发人员很难控制的。

(6) 用户界面的可用性。由于用户来源的广泛性，良好的用户界面是用户方便使用Web应用的最重要因素之一。因此，Web应用需要让未经专门培训的用户能够轻松正确地操作，用户界面应当直观、简单、美观、易用。学术界和工业界已进行了大量关于Web应用界面设计的研究，用户界面中重视网站导航和多媒体内容是Web应用和传统软件应用的一个显著区别。

(7) 内容的质量。内容是Web应用吸引用户的关键，同时也是传统的需求工程方法经常忽略的内容，Web应用开发人员需要更多地考虑内容的构建与维护。Web应用的内容具有的质量属性包含有准确性、客观性、可信性、相关性、现实性、完整性等。由于Web应用多媒体的特性，在设计Web应用时应考虑如何利用声音、视频等媒体来实现利益相关者的目标和需求。

(8) 开发人员缺乏经验。Web应用中使用的很多技术都很新颖，一方面这些新技术带来了Web应用开发效率的大幅提高；另一方面，由于开发工具、标准、语言、框架更新都很快，可能无法准确估算执行需求的代价，进而影响Web应用开发的进度和计划，导致项目延期或预算超标。而Web应用开发团队中的界面设计师、分析师等人员可能缺乏软件工程的背景，导致对Web应用所做的设计分析在工程上难以实现。

(9) 严格的预算与交付日期。大多数Web应用的开发方都是规模较小的创业团队，因此开发预算通常较为紧张。另外，由于Web应用通常都需要很快的上线速度，因此开发周期一般较短，需要对需求的优先级有明确的认定，对优先级高的重要需求优先进行开发。

(10) 与商业目标紧密相关。Web应用的商业目标是Web应用需求的重要组成部分，必须对商业目标进行详细的分析。Web应用不仅需要考虑提供用户所需的内容，迎合用户的目标，也必须能够满足Web应用自身的商业目标。

针对Web应用来讲，考虑到其应用的特殊性，可以将Web应用的需求分为功能需求、质量需求、系统环境需求、项目约束和发展需求等内容。

1) 功能需求

功能需求通俗来讲就是Web应用有什么用，需要做什么，能够为用户提供什么功能，能解决哪些问题。Web应用的功能需求是有层次性的，有核心功能要求和辅助功能要求，在开发过程中应该优先实现核心功能。功能需求常用UML中的用例图进行描述。

功能需求可被细化为以下几个方面的内容。

(1) 数据需求：也称为概念需求、内容需求或存储需求。数据需求确定Web应用的信息存储和管理方法，可包括如下方面。

① 内容采用何种表现方式(文本、视频、图片、表格、压缩包等)。

② 内容是由网站还是用户进行添加和维护。

③ 是否允许非原创的转载内容。

④ 是否允许用户对内容进行评论。

(2) 界面需求：也称交互需求，它定义一个Web应用如何与各种不同类型的用户交互。由于Web应用的用户群的知识结构、习惯、兴趣爱好等方面均有很大不同，因此Web应用应该让未经过正式培训用户能够根据自己的直觉轻松地使用。在需求获取过程中，开发原型系统是一种较好的用来获取用户的界面需求的方式。通过原型系统将用户对Web应用界面的期望反映出来并反复与用户交互，可以帮助开发人员设计出用户满意的Web应用。

(3) 导航需求：良好的Web应用应该让用户可以方便地在网站的不同频道与页面之间跳转，而不需要经常使用浏览器的前进/后退功能。

(4) 个性化需求：也称适应性需求，它描述一个Web应用在用户或环境变化下适应的能力，如Web应用支持用户自己选择不同的栏目展示在页面上，或用户自己设定页面背景颜色等。

(5) 事务性需求：也称内部功能性需求或服务需求，它表明一个Web应用必须进行的内部处理(后台处理)。事务性需求基本不需要考虑界面和交互方面的内容。

2) 质量需求

质量需求描述了服务的质量以及Web应用的安全性、性能及可用性方面的内容。主要的质量属性包括系统性能、可靠性、可用性、效率、可维护性和可移植性等。由于Web应用的性能严重依赖于网络情况，因此在分析质量需求时需要考虑网络带宽及网络延时等的影响。

(1) 性能：包括Web应用支持的最大并发访问数、响应时间、吞吐量、吞吐率等性能指标。不同类型的Web应用所关注的性能指标也有不同的侧重点。

(2) 可靠性：描述Web应用能在多长时间内保持正常工作状态。例如，对于一台关键的Web服务器，往往需要它一周7天每天24小时不间断地可用。通过备份、故障恢复等措施，可以提高Web应用的可靠性。

(3) 可用性：描述在要求的外部资源得到保证的前提下，Web应用在规定的条件、规定的时刻或时间段内处于可执行规定功能状态的能力。可用性是可靠性、可维护性的综合反映。

(4) 效率：描述Web应用利用计算资源的能力，在完成相同的计算任务时，Web应用占用CPU时间和内存空间等计算资源越少，说明应用的效率越高。换句话说，在相同的计算资源的条件下，完成的计算任务越多，效率越高。

(5) 可维护性：描述Web应用投入运行后对其进行维护的难易程度。理想状态下，Web应用在投入运行后可以在不停止服务的前提下，只需修改少量代码就可以实现新的功能需求或错误修正。影响可维护性的因素包括代码的可读性、可扩展性和可测试性等。需要注意的是，编写可维护性好的代码必然会加大开发者的工作量且降低开发速度，因此在可维护性和开发速度之间需要做出权衡。

3) 系统环境需求

系统环境需求描述一个Web应用如何嵌入到一个已有的环境中，以及它如何与遗留系

统、中间件等外部组件交互。

4）项目约束

对于项目的利益相关者来说，项目约束是需要多方协商解决的。典型项目约束包括项目预算、进度、技术限制、所采用的标准和开发技术等。

5）发展需求

Web应用需求经常会发生变更，因此，Web应用开发人员需要着眼于Web应用未来的发展。

由于Web应用需求工程的诸多特性，导致Web应用需求工程中经常会出现需求范围未界定、需求未细化、需求描述不清楚、需求遗漏和需求互相矛盾等问题。Web应用需求阶段不但要分析Web应用本身的功能和性能，还要对可能的用户群体进行分析和调查，并根据分析结果制定生成模型。

Web应用规模大型化、功能复杂化使得Web应用需求的开发和管理也日益复杂，需求工程自身也面临着需求获取与需求分析、需求表示、需求确认和验证等问题。针对这些问题，本书给出了一些Web应用需求工程的方法来管理Web应用的需求。以下将从Web需求获取、Web需求表示、Web需求确认与验证等方面进行具体的分析，并对主流的Web需求工具进行介绍。

3.2 Web需求获取

在获取需求的过程中，开发者与客户对项目中描述的需求必须达成统一的认识。需求获取应该集中在用户希望Web应用能够完成的任务上，而不应过于关注Web应用的用户界面。

需求不会凭空而来，Web应用开发人员从用户和客户那里收集所需要的信息，将这些信息从不同的信息源收集整理，信息源包括文档、遗留系统、问卷调查、面谈等形式。然后将需求以文档、图表等方式描述清楚，通过需求确认查看需求是否一致，检查是否存在需求错误以及是否存在没有定义的需求。这一过程迭代执行，在复杂的项目中会被执行多次。

由于Web应用的诸多特性，使得Web应用需求的获取相对传统的应用软件更为困难。因此，除了采用传统软件的需求获取方式外，Web应用需求人员需要掌握一些针对Web应用的需求获取方法。

3.2.1 需求准备

在进行Web需求获取之前，首先需要了解用户为什么会使用这个Web应用。

当创建Web应用的时候，很多Web应用开发成员认为用户会盯着每个Web页面，仔细阅读页面上的文字、图片等内容，熟悉页面的组织方式，然后再确定点击哪个链接。而事实上，这些开发人员假想的使用方式和用户实际使用方式之间存在着巨大差别。在大部分时间里，用户只是在每个页面上瞥一眼，扫过一些文字，点击第一个令他们感兴趣的或者大概符合他们目标的链接，而忽略页面上的很多部分。

用户在实际使用Web应用时有如下几方面的特点。

(1) 用户只是扫描页面。用户不是开发人员，不会过多地关注Web应用的细节，他们并不阅读Web页面的具体内容，而是扫描Web页面。通常用户使用Web应用只会寻找自己所需的有用信息，而不会花费太多的时间逐字逐句地阅读Web页面上的所有内容。因此，用户不希望花费太多时间和精力去阅读那些他们不需要的内容。

(2) 用户不做最佳选择。一般情况下，用户不做最佳选择，而是满意即可。对于用户而言，如果页面点错了，也不会产生严重的后果，顶多就是退回到上一页面而已。当用户在Web应用上做了一次错误选择后，通常只是点击几次后退按钮，只要找到所需要的内容他们就满意了。

(3) 用户不追根究底。用户只要Web应用能正常使用即可，不会过多地关注一些专业Web开发人员关注的专业问题(如链接错误、图像显示问题等)。

需求分析人员需要明白Web应用的需求来源于哪里。需求获取过程的起点是利益相关者的期望，在系统开发中，利益相关者可以是对需求有直接或间接影响的人或组织，重要的利益相关者是客户、用户和开发人员。Web应用典型的利益相关者包括内容作者、领域专家、可用性专家和(或)市场营销专家，这些利益相关者的期望一般比较多样性而且零散，例如以下的内容。

① Web应用应该在2011-6-1在线可用(客户约束)。

② Web应用必须应该至少能允许3000个用户并发访问(客户的质量目标)。

③ 使用Java EE作为软件开发平台(开发人员的预期技术)。

④ 所有用户数据应该被安全提交(客户的质量目标)。

⑤ 不同用户组可以为用户界面设置不同的布局(用户的质量目标)。

⑥ 用户能在3分钟内找到他想要的产品(用户的可用性目标)。

因为各自的立场不同，因此很可能出现不同的利益相关者的目标出现冲突和矛盾，如开发方所期望的预算超过了投资方愿意投入的资金总量等。在这种情况下需要各方面进行沟通和协商，确保对系统的目标是一致的。

3.2.2　需求获取方法

需求获取通常是由开发小组调查用户或潜在用户对系统的要求而得到，需求获取可以采取面谈、用例建模(Use Case Modeling)、头脑风暴(Brain Storm)、素描与故事板(Sketching and Storyboarding)、问卷调查和调查表(Questionnaire and Checklist)、联合应用开发(Joint Application Development, JAD)、原型化(Prototype)等方法。

1) 面谈

面谈是一种传统的常用方法，需求分析人员需要面对面地与需求提供人员就待开发的Web应用进行各种形式的讨论。通过面谈，需求分析人员能够理解到问题的症结所在，并且获得待开发Web应用的目标信息。常规的面谈过程包含4个步骤：识别面谈的利益相关者、面谈准备、进行面谈和将面谈结果记录为文档。该方法要求负责面谈的人拥有丰富的经验，同时要求负责面谈的人能够选择出最适合的人进行访问。

2) 用例建模

用例是一种描述系统需求的方法，常采用UML用例图的形式描述。用例是参与者(不仅仅是人，也包含其他软件系统)与系统的交互，代表了系统为其参与者所执行的有价值的

操作，用于表达系统的功能需求和行为。用例建模可以获取功能需求。用例图适合用做功能分析，但在对商业目标和与业务领域紧密相关的需求分析方面比较薄弱。

用例的目的是理解用户如何对系统进行操作，因此应该从参与者的角度来编写用例。用例方法的最大优点是能充分反映系统使用者的视角，因此，用户能充分地理解用例，也可以判断被捕获的软件需求是否能够满足他们真正的需要，从而加速双方在早期达成共识。建立用例模型的主要工作是识别角色、识别用例和描述用例。

3）头脑风暴

头脑风暴法通过组织团队会议，利用参会者的相互交流产生思维碰撞，利用组合效应进行创造性思维，很适合于用在需求获取的早期，尤其是第一次（或前几次）会议中采用。

采用头脑风暴法组织群体决策时，要集中有关专家召开专题会议，参会总人数建议不超过10人。主持者以明确的方式向所有参与者阐明问题，说明会议的规则，尽力创造融洽轻松的会议气氛。在头脑风暴会议中，由专家们自由提出尽可能多的方案。需要特别注意的是，参会者一般不发表反对意见，以免影响会议的自由气氛。

4）素描与故事板

该方法在Web应用的需求获取过程中常被界面设计人员使用，它通过绘制各个用户界面（页面）的示意图，让用户对系统的表现形式有感性了解。这些示意图被分组并通过链接连接起来，构成了所谓的"故事板"，可以很好地表示页面之间的导航结构。

5）问卷调查和调查表

问卷调查法是从多个需求提供人员中收集信息的有效方法，一般作为面谈法的补充形式。该方法需要设计包含大量问题的问卷，让调查人回答问卷中的问题。该方法的主要缺陷是分析人员需要对问题以及待构建的系统具有全面的了解才能准备好问卷调查表。

6）联合应用开发

联合应用开发方法由IBM于1977年提出，并在20世纪80年代首先投入使用。该方法可以替代面谈，它通过一连串的合作研讨会来获取用户需求，每次会议都会分析一些需求并且建立规范的文档记录，在每次会议的时候，小组会讨论不同的话题，然后以文档的形式记录下讨论经过，使得系统的需求变得更加明朗。该方法相比面谈、问卷调查等其他方法较为耗时，但获取的需求质量也较高。

7）原型化

由于Web应用是以页面的形式展现给最终用户，因此先开发出一个系统的原型（只设计页面不实现功能或实现简单的功能）可以让用户在直观上了解系统的展现情况，以便与需求提供者确认系统行为或者提出对系统行为的修改意见。原型法不仅可以用于需求的获取，还可以用于需求的确认与验证。

上述方法中，1）、2）、4）和7）较常用，在实际的Web应用开发过程中，也经常结合多种方法进行需求获取。

3.2.3 需求获取原则

Web需求分析人员除了需要掌握一些常用的需求获取方法外，还有一些常用的需求获取原则可以参考。

1）识别目标用户

不同Web应用由于自身定位的不同，有不同的目标用户。一个Web应用能否成功很大程度上要看其是否符合目标用户的需要和期望，这就要求Web应用必须有明确的目标用户群。需求分析人员需要明确Web应用的目标用户，即哪些人应该参与Web应用的需求分析活动，进而熟悉用户使用的习惯、方法和工具等内容。

2）全面考虑各利益相关者的意见

不同利益相关者的目标可能不同，甚至存在冲突和矛盾，而且需求通常是动态变化的。因此，在需求获取时，应充分考虑所有的利益相关者的目标，时刻注意解决和协调产生的冲突和矛盾。应该极力避免为了符合某方的利益而牺牲其他方面利益的情况。

3）了解系统所处的环境

很多Web应用并不是一个独立的系统，而是一个大的系统中的一部分。开发者需要对系统所处的大环境进行分析，并了解Web应用如何嵌入到这个大环境中。

4）明确调查内容

在需求分析的过程中，往往有很多不明确的用户需求。项目负责人需要调查用户的实际情况，明确用户需求。常见的调查内容包括如下几项。

（1）Web应用当前以及日后可能出现的功能需求。

（2）客户对Web应用的性能（如访问速度）的要求和可靠性的要求。

（3）确定Web应用维护的要求。

（4）Web应用的实际运行环境。

（5）Web应用页面总体风格以及美工效果（必要时用户可以提供其他类似的Web应用以供开发者参考，或者由开发者提供给用户以供选择）。

（6）主页面和次级页面数量，是否需要多种语言版本等。

（7）内容管理及录入任务的分配。

（8）各种页面特殊效果及其数量。

（9）项目完成时间及进度（可以参考合同内容）。

（10）明确项目完成后的维护责任。

调查结束以后，需要编写《用户调查报告》以记录调查内容。《用户调查报告》主要包括如下内容。

（1）调查概要说明：Web应用项目的名称、用户单位、参与调查人员、调查开始与终止的时间、调查的工作安排。

（2）调查内容说明：用户的基本情况、用户的主要业务、用户的信息化建设现状、Web应用当前和将来潜在的功能需求、性能需求、可靠性需求、实际运行环境、用户对新网站的期望等。

（3）调查资料汇编：将调查得到的资料分类汇总（如调查问卷、会议记录等）。

5）进行市场调研

需求分析人员需要进行市场调研活动。通过市场调研活动，获取Web应用背景资料，并分析其他相似Web应用的性能和运行情况。通过学习其他Web应用的核心定位与特色、实现功能和频道（如新闻、论坛、博客等）设置等内容，可以帮助项目负责人更加清楚地构想出自己开发的Web应用的大体架构和模样。但是，由于实际中时间、经费、公司能力所限，市场调研覆盖的范围有一定的局限性，在调研市场同类Web应用时，调研的重点应该放

在主要竞争对手的Web应用上。

为了便于读者理解，本书采用一个Web应用的案例——新闻系统。该系统的用户包括一般用户、注册用户、新闻撰稿人、新闻审核人员和系统管理人员，主要的功能包括新闻的提交、审核与发布、新闻管理与展示等。由于该系统最主要的功能是供内部的管理人员、新闻审核人员以及新闻作者使用，而不是让浏览网站的一般用户使用，因此可以采用与内容人员面谈、故事板等方法进行需求的获取。我们假设新闻系统应具有如下功能。

(1) 登录。根据填写的用户名和密码发送连接请求。连接成功后服务器对登录者的身份进行验证，根据登录者的权限进入不同界面。

(2) 新闻提交。新闻投稿人填写新增的新闻稿内容，向服务器发送增加新闻稿的请求，增加一条新闻稿。新闻稿录入的内容包括新闻标题、新闻类型、所属专题、新闻属性、新闻内容、来源等信息。新闻稿通过内容审核和格式审核后即可发布在系统中，同时管理员给新闻作者发送邮件通知。

(3) 新闻审核。包括内容审核和格式审核两项内容。内容审核主要由审核者对文章内容的合法性进行审核。格式审核由主编审核文章的格式。每项审核可能需要多名审核者参与。只有通过审核后，新闻稿才能发布在新闻系统中。

(4) 新闻发布。新闻管理员对通过审核的新闻稿，向新闻内容服务器发送发布请求，在内容服务器添加可浏览的新闻页面。

(5) 新闻管理。管理人员可以查询新闻(按照关键字、发布时间等方式)，更改新闻的内容，删除新闻，更改新闻所在栏目。

(6) 当日新闻一览。管理人员可以查询当日发布的新闻。

(7) 设置新闻的评论权限。设置是否允许注册用户对新闻进行评论。

(8) 用户管理。添加系统管理用户，分配相应栏目的操作权限，更改用户锁定状态(不允许处于锁定状态的用户登录)。

(9) 栏目管理。新闻栏目信息的增加、删除、修改，包括对特定栏目下的新闻设置不同的访问权限；新闻类别信息增加、删除、修改。

(10)日志管理。记录用户登录日志，查看和删除用户登录日志。

(11)系统管理。包括整个系统的数据备份、页面模板管理等内容。

3.2.4 敏捷需求获取

近年来，以极限编程(XP)和SCRUM为代表的敏捷软件工程方法日益受到软件业界的重视。相对传统的RUP等软件工程方法而言，敏捷方法属于轻量级方法，对软件需求的变化采取更为主动积极的处理方法，更重视人的作用，适合中小规模的开发团队，更符合Web应用的开发特点。

敏捷开发方法中一般采用故事板、用例建模、原型化等方法进行需求建模，而较少采用联合应用开发等较为复杂的方法。

敏捷软件工程中对需求获取有如下建议。

1) 明确权利与义务

高质量的需求源自开发人员和客户之间良好的沟通与合作。敏捷思想将开发人员和客户视为良好的合作伙伴关系。利益相关者之间相互理解、相互尊重，各方面都清楚各自的职

责并承担自己的义务,如有疑惑协商解决,以此减少后续因一方不愿意或不提供对方所需而产生矛盾。

开发者需要让客户了解需求具有不同的重要性和优先级以及忽视需求的危害性,保证需求分析人员与客户建立良好的合作关系,促使他们积极友善地参加需求阶段中的各项活动,让客户在制定项目计划时为需求阶段安排充足的时间。

2) 结对学习

开发者应该尽量让客户了解需求分析所需的知识,如Web应用安全性和可用性等。开发者也不能凭空想象需求,必须向客户学习其业务,深入客户的工作环境,了解客户的需求。极限编程的思想采用"结对"策略让一个客户和一个开发者结成对子,两人一起工作,共同讨论需求。这不仅有利于项目的成功,也能建立长期的友好合作关系。

3.3 Web需求分析、表示与管理

3.3.1 Web需求分析

需求分析包括提取、分析和审查获取到的需求,以确保所有的利益相关者都明白其含义并找出其中的错误、遗漏或不足。分析的目的在于得到高质量和具体的需求。需求分析经历的活动及需要遵循的原则如下。

1) 需求分析活动

由于Web应用的特殊性和行业覆盖的广阔性,以及需求分析的高风险性,所以,高质量的需求分析是Web应用成功的关键因素。现有的需求分析方法主要包括以下活动。

(1) 绘制系统关联图。关联图用于定义Web应用与其外部实体间的边界和接口的简单模型。对于本书中的新闻系统案例,由于基本不与其他软件系统发生交互,其外部实体主要是系统的用户。

(2) 创建用户界面原型。当开发人员或用户无法确定需求时,开发一个用户界面原型可以使Web应用具体化。用户可以对原型进行评估,使Web应用的参与者能更好地理解所要解决的问题,同时找出需求文档与原型之间的冲突之处。对于新闻系统案例,可以先绘制或实现简单的页面原型,让用户对系统有直观的认识,以便提出意见。

(3) 分析需求可行性。在成本和性能的约束下,分析每项需求实施的可行性,明确与每项需求实现相联系的风险,包括与其他需求的冲突、对外界因素的依赖和技术障碍。对无法实现或实现难度很大的需求,开发者需要和用户沟通,讨论是否需要去掉或修改这些需求。对于新闻系统案例,因为没有技术难度很大的需求,则主要需要考虑成本、时间上是否超出计划。

(4) 确定需求的优先级别。应用分析方法来确定用例、特性或单项需求实现的优先级别,以优先级为基础确定Web应用的各个版本将包括哪些特性或哪类需求。优先级很高的需求应在Web应用的第一个版本中实现,而优先级较低的需求可以在后续版本中实现。对于新闻系统案例,优先级高的功能是和新闻发布密切相关的功能,如新闻的审核与发布等。

(5) 为需求建立模型。图形化的需求分析模型能提供不同的信息与关系以帮助开发者发现不正确的、不一致的、遗漏的和冗余的需求。常用的模型可分为结构化需求分析方法和

面向对象需求分析方法两类。

结构化分析方法主要使用实体关系图、数据流图和状态转换图 3 种图形模型，分别进行数据建模、功能建模和动态建模。面向对象分析方法以用例模型为核心，采用 UML 为基础的类图、用例图、状态图、时序图和活动图等图形模型来对 Web 应用需求的各个方面进行刻画。

开发者可根据项目的实际情况选取方法，也可两种方法结合使用。

(6) 创建数据字典。数据字典是对 Web 应用所用到的所有数据项和结构的定义，以确保开发人员使用统一的数据定义。在需求阶段，数据字典至少应包含和客户相关的数据项以确保客户与开发小组是使用一致的定义和术语，避免出现二义性。

2) 需求分析原则

需求分析过程是一个提炼用户需求的过程，需要将用户的需求描述提炼成 Web 应用开发中可以为所有业务相关人员都能理解的形式，对于这一复杂过程，一般需要遵循一定的原则。

(1) 注意需求描述用语。开发者应使用符合客户语言习惯的表达方式，因此开发者需要了解客户使用的业务术语，但客户不一定需要懂得计算机行业的术语。对于 Web 应用来说，如果用户为普通互联网用户，则不会有大量的业务术语。但如果是类似于本书案例中的新闻系统，则开发者需要了解新闻发布流程中常用的业务术语。

(2) 了解客户业务。只有分析人员更好地了解客户的业务及目标，才能使产品更好地满足需要。这将有助于开发人员设计出真正满足客户需要并达到期望的优秀 Web 应用系统。客户可以邀请开发和分析人员观察用户的工作流程。如果是旧系统切换到新系统，则开发人员和需求分析人员应使用一下目前的旧系统，有助于了解目前系统的工作流程，以便在新的系统中进行改进。如果是以前没有的系统，则应该对之前人工的工作方式进行分析，抽取出业务流程以便在系统中实现。

(3) 描述产品的非功能特性。客户对 Web 应用的非功能特性的描述通常会比较主观。例如，客户有时要求产品"界面友好、运行高效"，但对于开发人员来讲，这些要求太主观、太抽象，其实并无实用价值。正确的做法是，分析人员通过询问和调查了解客户所要的"友好"、"高效"所包含的具体特性，分析特性之间的影响，在性能代价和所提出解决方案的预期利益之间做出权衡，以确保做出合理的取舍。

(4) 评估需求变更代价。如果用户需要对需求进行变更，则开发人员应通过分析给出一个真实可信的评估，包括成本、开发时间的变更等，以便用户正确了解需求变更带来的影响。需要注意的是，开发人员不能由于不愿实施变更而随意夸大评估成本。

(5) 客户参与。作为系统的最重要的利益相关者，客户在需求工程的过程中扮演着非常重要的角色，客户应该尽可能地参与到整个需求工程的过程中。客户首先需要向开发者详细地介绍 Web 应用应该实现的业务逻辑，在开发者形成需求文档后用户也应该积极地对文档的内容提出修正意见，并去掉不合理的需求。

3.3.2 Web 需求表示

在完成需求获取和需求分析的工作后，应将需求描述清楚并表示出来。

根据需求表示的详细程度和正规程度，需求可以用多种方式表示，如需求故事

(Stories)、条目化需求(Itemized Requirements)、格式化需求规格说明(Formatted Specifications)、正规需求规格说明(Formal Specifications)等。下面分别对这几种Web需求的表示方法进行介绍。

1) 需求故事

需求故事是以口语化的方式描述需求所包含的内容,通常用于开发者和用户之间的交互。

Web应用开发中最常用的需求故事方法是极限编程中提出的用户故事(User Stories)。用户故事以用户熟悉的语言编写,描述了对于用户而言系统能解决的问题和完成的工作。新闻系统中管理员登录的用户故事可如图3.1所示。

> 一名管理员在登录页面输入其用户名、密码和验证码,并单击"登录"按钮。如果管理员输入的用户名、密码和验证码三者均正确,则登录成功并根据该管理员的权限跳转到后台管理页面。否则给出登录失败的提示,并清空用户输入的内容,更换新的验证码。

图3.1 用户故事示例

2) 条目化需求

条目化需求以自然语言编写,包含多项具体的需求,其中每一项需求都有一个唯一的标识符来与其他需求区别开。IEEE/EIA J STD-016标准中的数据项描述(Data Item Description,DID)是条目化需求的典型例子。

3) 格式化需求规格说明

格式化需求规格说明采用严格定义的语法,但仍然允许用自然语言进行描述。最常用的格式化需求规格说明是UML中的用例描述。

4) 正规需求规格说明

正规需求规格说明不是采用自然语言写成,而是采用形式化规定的语法和语义。由于Web应用的需求通常会发生频繁变更,因此绝大多数情况下不考虑使用正规需求规格说明。

针对Web应用需求经常发生变更的特性,通常不需要对需求进行非常精确的描述,也不需要进行形式化验证。开发者的重点应该放在如何把需求向不了解Web开发的用户解释清楚。由于Web应用通常追求快速灵活的开发模式,因而需求故事、格式化需求规格说明等方法比较适合Web应用开发。

对于Web应用而言,需求分析阶段需要编写软件需求规格说明作为整个需求分析活动的结果性文档。需求规格说明阐述一个Web应用必须提供的功能和性能以及它所要考虑的限制条件,是Web工程中项目成员主要的参考文档。它不仅是系统测试和用户文档的基础,也是项目规划、设计和编码的基础。

软件需求规格说明是用户、分析人员和设计人员之间进行理解和交流的手段。测试人员可以根据软件需求规格说明中对产品行为的描述,制定测试计划,设计测试用例和测试过程。文档人员根据软件需求规格说明和用户界面设计,编写用户手册等文档。软件需求规格说明指导着整个Web应用的开发过程,已通过评审的需求规格说明需要进行变更控制。

对于Web应用而言,一份完整的软件需求规格说明一般需要包含以下内容:

- Web应用功能;

- 初步的 Web 应用用户界面；
- Web 应用运行的软、硬件环境；
- Web 应用性能要求；
- 确定 Web 应用维护的要求；
- 确定 Web 应用空间租赁要求；
- Web 应用页面总体风格及美工效果；
- 主页面及次页面大概数量；
- 管理及内容录入任务分配；
- 各种页面特殊效果及其数量；
- 项目交付时间及进度安排表；
- 明确项目完成后的维护责任。

在较大规模的 Web 应用项目中，开发者通常采用标准的软件需求规格说明的模板。读者可参考 IEEE 830 1998 或其他标准所提供的模板，并根据 Web 应用的实际情况对文档内容进行定制。

IEEE 830 1998 标准模板包含引言、综合描述、外部接口需求、系统特性、其他非功能需求、其他需求 5 个方面，其中针对 Web 应用实际情况，需要特别注意文档中的如下内容。

(1) “综合描述”中的“运行环境”内容。不仅需要说明 Web 应用后台服务器所运行的硬件平台、操作系统和版本，还应详细说明 Web 应用支持的浏览器类型和版本。

(2) “外部接口需求”中的“用户界面”内容。需要描述 Web 应用中用户界面的风格，并注明 Web 应用支持的屏幕分辨率。

(3) “其他非功能需求”中的“性能需求”内容。需要注明 Web 应用支持的最大用户并发数和实时性的需求。

3.3.3 Web 需求管理

需求管理开始于需求获取及分析完成后，其目的是管理需求的变更和研究需求变更将带来的影响。需求管理是一种用于查找、记录、组织和跟踪系统需求变更的系统化方法，不仅要管理需求变化过程，还需要维护需求变化后的一致性和完整性。有效需求管理的关键在于维护每项需求的详细说明、每种需求类型所适用的属性以及与其他需求和其他项目工件之间的可追踪性。

1. Web 需求管理的意义

需求管理对于 Web 应用开发意义重大，主要表现在以下几个方面。

(1) 可以更好地控制复杂的系统。如果开发者缺少对系统行为及需求变化的认识，会导致系统失去控制。需求管理有助于系统开发人员清晰理解 Web 应用的功能和完成时间的要求，也能将项目资源在需求优先级和影响的基础上进行合理的分配，需求变更的影响也能够得到很好的处理。

(2) 提高软件产品的质量和客户的满意度。高质量的软件产品必然是建立在开发者和测试者对软件需求的正确理解之上，软件产品是否符合需求的描述是决定软件质量的关键因素。

(3) 降低项目成本。需求分析阶段所产生的错误，其更正代价是最大的。在项目早期发现并解决这些错误可以减少不完善的需求对未来项目开发的影响。

(4) 促进用户与开发者的沟通。需求管理要求用户积极参与，可以促进用户与开发者的沟通，让用户与开发者对需求的理解保持一致。

(5) 使项目的过程更加符合标准。大多数软件和过程改进的标准都涉及需求管理。业界最常用的能力成熟度模型(Capability Maturity Model，CMM)就将需求管理作为软件质量改进的第一步。ISO9000 标准也都要求公司执行需求管理。

2. 需求管理的内容

Web 需求管理包括需求的变更管理、文档版本控制、状态跟踪和需求跟踪 4 个部分。

1) 变更管理

需求不可能是完备的，随着项目的开展和时间的推移，用户需求发生变更是不可避免的。所谓需求变更是指在 Web 应用需求基线已经确定之后，又要添加新的需求或进行较大的需求修改。Web 应用由于其自身的特点，其开发的过程中通常会产生大量的需求变化。

从坏的方面讲，需求的变更可能会增加新的需求，增加项目开发工作量，如果不允许用户变更需求或者没有一个规范的需求变更控制过程，都会给用户和开发者带来很大的项目风险，严重的可能导致项目开发的失败；从好的方面讲，需求出现变化可能会带来新的商业机会，让开发者发现新的更合理的需求。绝大多数情况下，Web 应用开发者需要采用适当的需求管理过程来对需求的变更进行跟踪。

如果需求变更处理不好，将给 Web 应用带来不良的影响，可能产生如下后果。

(1) 影响 Web 应用质量及开发进度。若没有完整的需求跟踪手段，在评估变更影响时难免会遗漏需求发生变化所影响的某些环节，可能在实施变更过程中产生难以察觉的错误，影响 Web 应用的质量并拖后开发进度。

(2) 影响文档和代码的一致性。文档是维护 Web 应用的重要依据。在处理需求变更的过程中，如果没有采用规范的流程保证需求变更的评估与实施，很可能会造成文档与 Web 应用不一致，导致系统维护困难。

(3) 影响开发人员与用户的合作关系。需求变更的实施是用户和开发人员相互协作的过程。开发人员和用户在是否采用变更问题上常常产生分歧，如果处理不当则可能会破坏相互之间的信任，影响项目开发进程。

有效地变更管理时，需要对变更带来的潜在影响及可能的成本费用进行评估。变更控制委员会与关键的项目风险承担者需要进行协商，以确定哪些需求可以变更，而哪些需求不宜发生变更。同时，无论是在开发阶段还是在系统测试阶段，还应跟踪每项需求的状态。建立起良好的配置管理方法或采用相关的需求变更管理工具能为提高需求管理的效果提供帮助。

需求变更的常见流程如下。

(1) 确定需求的基线。通常会以用例作为需求基线，在用例确认之后的任何需求改变，都需要经过需求变更流程，而没有经过需求变更流程的需求将不被认可。

(2) 项目经理接收到需求变更的要求。需求变更的提出者可以是项目中的任何相关人员，包括项目经理、客户、用户、开发人员和测试人员等。

(3) 项目经理评估需求变更。项目经理可以召集相关人员讨论需求变更的合理性、可行性、实施的代价以及对项目的影响。项目经理作为项目的负责人，对项目的成功负有主要的责任，因此需求变更的决策者应该由项目经理承担。对需求变更的评估也需要用户的参与。

(4) 记录需求变更。需求变更确认后由专人将需求变更记录下来，将需求变更的原因等信息通知给项目中所有成员。其中用户方代表、需求分析师、测试人员和相关开发人员与需求的变更是紧密相关的，他们必须知晓并认可此需求变更。

(5) 确认需求变更。相关人员接收到确认的需求变更后，需求分析人员修改项目文档中的相关内容，主要包括需求规格说明书和用例文档。接下来开发人员修改代码中的相关部分，将修改后的代码作为新的程序版本。应采用版本控制工具保留每次更改过的版本，而不能用新的版本覆盖旧的版本，以便跟踪到需求变更过程中所带来的工作调整，并防止日后新版本出现问题时无法恢复到旧版本。版本控制工具支持多人团队使用，避免了不同开发者修改同一份代码所可能导致的代码不一致的弊端。

(6) 需求冻结。项目开发越是进入到后期，需求变更对项目的影响就越大，所以在项目开发到一定程度时会进入需求冻结阶段，不再接收需求的变更。

2) 版本控制

版本控制也称版本管理，是对开发过程中代码和文档的版本进行记录和保存，通过合适的规则来创建新版本，删除旧版本，合并已有版本，等等，它是管理项目文档必不可少的内容。通常采用 SVN、CVS、Git 等版本控制工具来进行需求规格说明的版本控制。

需求文档的每次修改都应作为一个新的版本，小组内每个成员必须能够得到需求的当前版本，必须清楚地将变更写成文档，并及时通知到项目开发所涉及的人员。建议指派一个人专门负责更新需求，并确保只要需求有所变更就相应地改变其版本标识号。

每一个公布的需求文档的版本应该包括一个修正版本的历史情况，即已做变更的内容、变更日期、变更人姓名以及变更原因，可以考虑给每个需求标记上版本号，当修改满足需求后就增加版本号。

3) 状态跟踪

在整个开发过程中，跟踪每个需求的状态是需求管理的一个重要的方面。通常可将需求的状态归类到如下 9 种状态中的一种。

(1) Open(开放)：对于原始需求或接收到的正式需求，在未正式进行需求分析之前的需求状态统一定义为"Open"状态。

(2) Analyzed(已分析)：对需求状态为"Open"的需求，若已完成需求分析过程，但还未正式通过需求评审前，其状态统一定义为"Analyzed"状态。

(3) Reviewed(已评审)：对需求状态为"Analyzed"的需求，若已正式通过需求评审，但还未完成测试，或测试结果为不合格之前，其状态统一定义为"Reviewed"状态。

(4) Resolved(已编码)：对需求状态为"Analyzed"或"Reviewed"的需求，若已完成需求设计和编码，且已通过单元测试，其状态统一定义为"Resolved"状态。

(5) Passed(已完成)：对需求状态为"Resolved"的需求，如果已通过正式测试，其状态统一定义为"Passed"状态。

(6) Unresolved(未测试)：对需求状态为"Resolved"的需求，如果未经过正式测试，其

状态统一定义为“Unresolved”状态。

(7) Closed(已结束)：对需求状态为“Resolved”的需求，若该需求已正式上线商用，且得到客户和项目团队的共同认可后，其状态统一定义为“Closed”状态。

(8) Cancel(已取消)：当某些需求被取消时(包括上线前取消和上线后取消)，其需求状态统一定义为“Cancel”状态。

(9) Failed(需修正)：对需求状态为“Closed”的需求，若该需求在Web应用上线后发现问题或存在缺陷，需要对其进行修正时，其需求状态统一定义为“Failed”状态。

通常，项目开发者通过周期性地检查各状态类别的需求在需求总数中所占的比例来对项目进行监控。

4) 需求跟踪

需求跟踪是需求管理的基础，同时也是需求变更控制的基础。需求跟踪通过比较需求文档与后续工作成果之间的对应关系，确保软件项目是依靠需求文档进行开发的。

需求跟踪包括编制每个需求同系统元素之间的联系文档，这些元素包括其他类型的需求、体系结构、其他设计部件、源代码模块、测试和帮助文件等。当需求发生变化时，使用需求跟踪可以确保不遗漏受到影响的系统元素，从而保证需求变更的正确实施，达到改善产品质量、降低需求变更带给项目的风险的效果。需求跟踪的目的是建立与维护“需求—设计—编码—测试”之间的一致性，确保所有的工作成果符合用户需求。

需求跟踪包括如下几个方面的内容。

(1) 建立与维护需求跟踪表。需求跟踪表记录软件需求与各个软件工程活动的阶段结果之间的映射关系，这种关系可能是“一对一”、“一对多”或“多对多”的。以需求跟踪表为基准，可以实现正向、逆向以及双向的需求跟踪。正向跟踪检查需求文档中的每个需求是否都能在后续工作成果中找到它的对应点。逆向跟踪检查涉及文档、代码、测试用例等工作成果是否都能在需求文档中找到对应的出处。双向跟踪是正向跟踪和逆向跟踪的结合。

(2) 一致性检查。使用需求跟踪表的优点是很容易发现需求文档与后续工作成果之间的不一致。这些不一致包括后续工作成果未实现或未正确实现需求文档中的某些需求，后续工作成果实现了需求文档中不存在的需求，等等。当发现存在不一致时，应记录在需求跟踪报告中，并通报相关责任人。

(3) 消除不一致。相关责任人针对需求跟踪报告中的不一致，给出解决的措施和计划，并提交给项目管理者以便记录到需求跟踪报告中。不一致消除后，应提醒项目管理者更新需求跟踪表。

3.4 Web需求确认与验证

Web应用需求的确认与验证是验证需求是否正确，是否合理，是否存在着与实际情况不符的问题，并解决这些问题。需求的确认与验证必须要有Web应用实际用户参与。

通过需求确认与验证，可以确保需求说明准确、完整地表达。如果需求规格说明书不完善，开发人员可能在阅读需求规格说明书时觉得需求是正确的，但实现时发现在技术上无法实现。此外，当以需求规格说明书为依据编写测试用例时，开发人员可能会发现某些需求的说明存在二义性。

需求的确认与验证是一个反复迭代的过程，在整个Web应用的开发过程中可能需要进行多次。通过“确认与验证→修正需求→再次确认与验证→再次修正需求……”的反复过程，才能最终得到用户与开发者一致认可的反映Web应用真实要求的需求。

最终得到的需求采用需求规格说明书形式的文档进行表示，一份合格的需求规格说明书应具有如下5个特点。

(1) 正确。需求规格说明对系统功能、行为、性能等的描述必须与用户的期望相吻合，能反映出用户的真正要求。

(2) 完整。需求规格说明应该包括软件要完成的全部任务，不能遗漏任何的需求信息。

(3) 一致无二义。需求规格说明对需求的描述不能存在矛盾之处。需求规格说明中的描述对所有人都只能有一种明确统一的解释，应尽量把每项需求用简明的易于用户理解的语言表达出来。

(4) 可修改。需求规格说明的格式和组织方式应保证易于进行后续的修改。建议使用版本控制工具等软件工具管理软件需求规格说明。

(5) 可跟踪/可验证。可跟踪性意味着每项需求都能与其对应的来源、设计、源代码和测试用例联系起来，这样便于在开发阶段跟踪各项需求是否已经实现。可验证性意味着每项需求都应该有明确的方法来验证其是否已经实现。

从上述特点可知，需求确认与验证的目标如下。

① 软件需求规格说明正确描述了预期的系统行为和特征。

② 需求是完整的和高质量的。

③ 任何对需求的看法是一致的。

④ 需求为继续进行产品设计、构造和测试提供了足够的基础。

需求确认与验证常用的形式包括如下6种。

(1) 评审和走查(Review or Walkthrough)。一般的评审分为客户评审和同行评审两类。在评审中，客户代表和开发人员代表各自检查需求文档，并开会讨论双方关心的问题，会议内容如下。

① 评审Web应用的目的和目标。

② 将需求和目标、目的相比较，以确定所有的需求都是必要的，并且不存在被遗漏的需求。

③ 评审Web应用的运行环境，检查Web应用与所有其他系统之间的接口，并检查其描述是否是正确的和完备的。

④ 客户代表评审需求是否准确地反应了客户的需要和意图。开发人员代表评审需求所包含的功能和约束，以证实它们是可实现的。

⑤ 评价开发过程中或Web应用运行过程中可能存在的风险，讨论和比较各种可选方案，并就将要使用的方法达成某种一致。

每当发现了问题，就将问题记录在文档中，然后确定其原因，接着需求分析员负责解决该问题。在设计开始之前，必须解决需求中存在的问题，开发人员可能需要构造原型系统来研究可行性约束，并和客户一起对需求达成一致意见。

(2) 审计(Audit)。在审查文档的过程中执行的检查，将结果与过程开始之初预先定义的列表进行比较。

(3) 需求跟踪矩阵(Requirement Traceability Matrix,RTM)。RTM 是表示需求和其他系统元素之间联系的最常用的方式之一,可以检测需求中的不一致和未覆盖的问题。

RTM 分为纵向跟踪矩阵和横向跟踪矩阵两类。纵向跟踪矩阵包含三种关系：需求之间的派生关系,描述了客户需求到产品需求的关系；实现与验证之间的关系,描述需求到设计、需求到测试用例的关系；需求的责任分配关系,描述需求由谁来实现。横向跟踪矩阵表明需求是由谁来实现。

有多个角色参与建立跟踪矩阵活动,如需求获取人员负责客户需求到产品需求的 RTM 建立,设计人员负责需求到设计的 RTM 的建立,测试用例的编写人员负责需求到测试用例的 RTM 建立,等等。质量保证人员负责检查是否建立了 RTM 及是否所有的需求都被覆盖了。

RTM 主要有两方面优点。

① 在需求变更、设计变更、代码变更、测试用例变更时,RTM 是目前最有效地进行变更波及范围影响分析的工具。如果不借助 RTM,则发生上述变更时,往往会遗漏某些连锁变化。

② RTM 可以验证需求是否已得到实现。借助 RTM,可以跟踪每个需求的状态,进而判断是否设计了需求,是否实现了需求,是否测试了需求。

表 3.1 说明了每个需求项与一个或多个设计、代码和测试元素的关系。设计元素可以是模型中的对象,例如数据流图、关系数据模型中的表单、对象类等。代码模块可以是类中的方法,源代码文件名、过程或函数。也可以根据 Web 应用开发的实际需要增加更多的列项,例如可以添加一列名为“帮助文档”的信息,用以记录某个需求项与帮助文档的对应关系。

表 3.1　需求跟踪矩阵

需求项	设计文档	代码	测试用例
需求项 1	设计元素 1	代码模块 1	测试用例 1
需求项 2	设计元素 2	代码模块 2	测试用例 2
…	…	…	…
需求项 n	设计元素 n	代码模块 n	测试用例 n

(4) 原型验证。用户通常不善于精确描述自己的业务需求,系统分析员需要借助白板、白纸等沟通方式,帮助用户清楚表述需求。然后,开发一个用户界面原型,以便用户确认需求。如果需要的话也可以实现一些简单的功能。原型通常只执行部分功能性需求,而不提供完整的用户界面。

和软件原型一样,Web 应用原型也有许多类型,从其内容来看,它既可以注重用户界面,也可以注重某些算法功能和效果；从其表现形式来看,可以是电子的,也可以是书面的；从其评价后的处理来看,可以将其抛弃,或继续将其进化为最终产品的一部分。一般来说,需要根据实际情况决定采用什么类型的原型,主要考虑的因素包括项目特点、开发人员水平、支持原型开发的工具和技术等。

原型验证虽然很直观,但也存在两大风险。一是用户看到一个正在运行的原型便以为产品即将完成,用户可能会认为只对原型进行一些修改就可以交付。由于原型没有考虑软

件的总体质量和可维护性，交付原型往往造成"欲速则不达"的情况。因此，必须让用户明白他们观察到的只是一个原型而不是最终的系统。二是开发人员为了快速构造原型，可能会采用不合适的操作系统，中间件或程序设计语言，也可能使用一些效率低的算法。在一段时间的开发之后，他们往往已经习惯了这些选择，于是便在系统中保留了这些不理想的选择。因此，开发人员需要分清原型和真实应用之间的区别，对真实的应用不能一味沿用原型的开发方式。

(5) 模型验证。当系统需求已经采用某种结构化或形式化的表示机制来刻画的时候，可以采用已有的模型验证技术来进行需求的确认与验证。

模型验证一般有三个目的。

① 证明每个独立的模型是自包含的，即模型应该包含所有必需的信息，并且模型的不同部分之间应该没有冲突。

② 如果系统有多个模型，证明它们是内容一致和外部一致的，即多个模型中涉及的实体/概念应该有相同的定义，模型之间的接口也应该一致。

③ 证明模型精确地反映了系统需求相关者的真实需求，这是模型验证中最困难的任务，即需要说明模型定义的系统确实是用户需要的系统。

(6) 需求建模测试。需求的建模包括把需求转换成图形模型或形式化语言模型。需求的图形化分析模型包括数据流图、实体关系图、状态转化图、对话图和类图等。

这些图形化模型通常需要借助一定的CASE(Computer-Aided Software Engineering，计算机辅助软件工程)工具来利用自动化分析工具本身提供的检测手段对需求进行测试，包括需求描述上的完整性检查、需求项之间的不一致性检查等。同时，使用CASE工具有助于达到需求的质量特性，包括有效性、一致性、可靠性、可用性、正确性、可维护性、可测试性、可扩展性和可重用性等。

Web应用需求的确认与验证保证了需求的完整性与正确性，为Web应用的开发指明了确切的方向。

3.5 Web需求工具

对于项目管理，除了需要具有合理的方法以及人员的干预以外，也很有必要使用相关的需求工具来提高分析人员和开发人员进行需求管理的效率，来更好地完成需求工程的任务。

需求工具可以分为以数据库为核心和以文档为核心两大类。

以数据库为核心的需求管理工具以CaliberRM和DOORS为代表，它们把所有的来自不同文档中的需求属性和跟踪能力信息存储在数据库中。需求的文本描述被简单地处理为必需的属性，工具还提供每个需求与外部文件如Word文件、Excel文件、图形文件等相联系的功能，通过这些外部文件对需求说明进行补充。

以文档为核心的需求管理工具以RequisitePro为代表，它允许选择将整份需求文档存储在数据库中，加强了以文档为核心的处理方法的能力。RequisitePro允许用户定义属性和跟踪能力联系链，类似以数据库为核心的工具，它也提供了同步数据库和文档内容的机制。

1) CaliberRM

CaliberRM是Borland公司的一款基于Web的支持多人协作的需求定义和管理工具，

可以帮助实现分布式的开发团队协作，从而加速交付应用系统。CaliberRM 协助团队成员沟通，可以减少错误和提升项目质量，也有助于更好地理解和控制项目。CaliberRM 提供集中的存储库，可以让全体成员保持同步，能够帮助 Web 应用项目团队在早期及时澄清项目的需求。此外，CaliberRM 可以和其他的对象建模工具、软件配置管理工具、项目规划工具、分析设计工具以及测试管理工具很好地集成。

可在 http://www.borland.com/us/products/caliber/index.html 下载 30 天体验版。

2) RequisitePro

RequisitePro 是 IBM 公司 Rational 产品线中的需求和用例管理工具，能够帮助项目团队改进项目目标的沟通，增强协作开发，降低项目风险，以及在部署前提高应用程序的质量。RequisitePro 支持与 Microsoft Word 集成，让开发者在进行需求的定义和组织时使用熟悉的软件环境。它提供数据库与 Word 文档的实时同步能力，便于进行需求的组织、集成和分析。RequisitePro 提供了详细的可跟踪视图来显示需求间的父子关系，以及需求之间的相互影响关系。通过软件导出的 XML 格式的项目基线，可以比较项目间的差异。

可以在 www.ibm.com 下载 15 天体验版。

3) DOORS

DOORS 是 IBM 公司 Rational 产品线中的多平台的需求管理解决方案，它提供覆盖整个开发生命周期的项目可视性、可跟踪性和需求管理功能，通过优化整个组织和跨供应链的需求通信、协作和验证，达到降低成本、提高效率和改进质量的目的。

DOORS 是基于数据库的工具，所有的需求数据都统一存储到单一的数据库中。DOORS 支持多种操作系统，例如可以从 Windows 的客户端访问位于 UNIX 或 Linux 上的 DOORS 数据库。DOORS 为用户提供简单、强大而完全的安全机制。DOORS 具有灵活的权限控制，包括只读、修改、创建、删除、管理等 5 个级别，权限控制可以针对每个用户在每个数据库、项目目录、文件、需求项、属性上实施。

可以在 http://www.ibm.com/software/awdtools/doors/ 查看演示。

上述三种主流的商用工具的特性比较如表 3.2 所示。

表 3.2 三种主流的商用工具的特性比较

功能项	子功能项	CaliberRM	RequisitePro	DOORS
需求的描述与排序	需求的描述	支持	支持	支持
	需求的分解	支持	支持	支持
	对需求进行排序	支持	支持	支持
需求依赖关系的支持	需求间建立依赖关系	支持	支持	支持
	需求非直接的依赖关系	支持	支持	不支持
	需求变化时标记可疑关系	支持	支持	支持
版本及基线的支持	需求变化时产生新的版本号	支持	支持	不支持但保存历史更改记录
	将需求发布成基线	支持	不支持	支持
	版本间的比较	支持	不支持	支持历次改动间的比较

续表

功能项	子功能项	CaliberRM	RequisitePro	DOORS
工作产品链接的支持	链接工作产品	支持	不支持	支持
	需求变化时标记相关工作产品为可疑	不支持	不支持	不支持
对交流的支持	邮件提醒	支持	支持	支持
	异步异地	支持	支持	支持
	不同角色间的讨论	支持	支持	不支持
对流程的支持	需求变更的过程	支持	支持	支持
	变更管理过程的定制	不支持	不支持	支持(采用脚本语言)
其他特点	项目开发可扩展性		需求的数据存在数据库中，与需求相关的上下文信息存放在Word文档中	一个数据库能够同时支持多个不同项目，便于复用和共享

4）其他商用产品

TestTrack RM是Seapine公司的一款需求管理产品。它包含计划、工作流、跟踪、审查、变更管理、报告等功能。可以在http：/www. seapine. com/ttrm. html 下载30天体验版。

DevSpec是TechExcel公司的一款需求定义与需求管理工具，基于预定义的流程规则，团队可以使用DevSpec全程跟踪每一个产品功能和特性的开发过程，通过需求管理来驱动研发与测试流程。DevSpec支持Scrum、迭代开发、瀑布开发等多种开发模型，并可以与Microsoft Office、Adobe PDF集成。内置的高质量报表功能，可以创建和定制包括需求 功能列表、需求变更分析与预测以及与需求/规范点相关的研发数据报表等在内的各类报表。可以在http://www. techexcel. com. cn 进行在线体验。

5）开源产品

除商用产品外，用于需求工程的开源产品也很多，如Trac和Redmine。

Trac是基于Web的软件项目开发跟踪工具，它最大的特点是支持众多的插件，对WiKi也有很好的支持。可以在http：/trac. edgewall. org/ 下载到Trac及其源代码。

Redmine是另一款基于Web的项目开发管理工具，用Ruby on Rails框架开发，支持多项目、多数据库、多语言，对项目开发中的文档管理、进度管理、权限控制、成员交流等都有很好的支持。可以在http://www. redmine. org/进行在线体验及下载。

需求管理工具可以保存与需求相关的信息，维护需求文档的版本记录，定义跟踪能力联系链以帮助分析需求变更的影响，并具备与其他软件开发工具的接口。尽管需求管理工具简化了手工需求管理的工作，但用户对需求的理解仍然是需求获取成功的关键因素。另外，商业的需求管理工具的价格一般较高，购买时需要权衡利弊。

3.6 总结与展望

本章主要讨论Web应用需求工程的特性。需求定义的好坏直接影响了开发者对Web应用的理解程度，是影响一个Web应用成功的重要因素。本章首先介绍了Web需求的特性，接下来分析了Web需求获取、Web需求表示、Web需求验证与确认以及相应的Web需求工具。通过了解这些内容，Web开发者能够更好地获取到有用的Web系统信息，最终开发出满足用户需求的Web产品。

由于Web应用的特性，其需求在开发过程中变更较为频繁，且对开发的时间要求一般较紧，项目的生存期也较短。因此，针对Web应用开发，其需求工程未来的主要发展趋势主要包括：开发和使用系统的边界越来越模糊，需求和架构之间的集成越来越紧密，新的支持分布式需求工程的工具逐步出现，Internet的开放性使得Web应用中使用的各种开发系统的需求工程越来越重要。

第4章 Web应用建模

Web应用在给用户带来巨大方便的同时也给开发人员提出了一个不可忽视的问题，就是如何降低Web应用开发的风险和保证其质量。基于模型的方法提供了比特定(Ad hoc)的Web应用开发更好的方法，以解决类似在前面章节提到的需求表达不充分、不准确以及没有需求规格文档等问题，降低Web应用开发风险。从静态和动态两方面为Web应用构建内容、超文本、展示层模型，以及个性化的适应性模型。Web应用的内容模型和传统软件系统类似，旨在捕获底层信息和应用逻辑，同时考虑超文本特性；超文本模型在内容基础上体现所有导航特性；展示模型则将超文本结构与链接和网页相对应；个性化的适应性模型则关注用户、时间、位置以及设备等上下文信息。

本章主要讨论现有的用于Web应用建模的方法和工具及其特性，读者可以从中选择合适的建模方法和工具。这些方法主要集中在基于模型的开发和代码生成工具方面。

4.1 Web应用建模特性

要在院子里搭建一个狗窝，只要你有一双灵巧的双手，以及砖(石块)、水泥等原材料和类似锤子等简单工具，就可以根据自己的喜好而完成。而要盖房子，则一个人的力量就无法完成；教堂则需要考虑更多的因素，尤其是独特的精神方面的因素；类似鸟巢、水立方等建筑物则更需要有设计、建立可视的模型并进行评审等系统化的工作。

模型是现实世界的抽象表达，用于发现领域中的对象(概念)，并为这些对象分配职责。通过简化一些细节，模型可以帮助理解系统。如何选择建模对象对理解问题和提供解决方案有重大影响。模型又是人们思维的工具，可以降低复杂性，并把决策进行文档化记录。模型又是项目有关的不同利益相关者之间有效的沟通工具。模型是模型驱动开发方法的核心。

建模的目的是为要实现的系统提供足够详细的规格说明，它有助于对系统进行可视化，帮助开发者正确地构建系统，使开发工作进展得更快。建模是一个必需的过程，也是Web应用工程化所必经的阶段。建模过程的产出是表达系统相关方面的简化的、易于理解的模型。

计算机科学领域已经使用建模的方法开发某些传统软件。其中，建模的对象是要创建的应用程序，涉及的范围跨越如下三维。

(1) 层，包括应用逻辑和用户界面层，封装了“什么”和“如何”实现应用。

(2) 方面，包括结构(如对象及其属性以及与其他对象的相互关系)和行为(如函数和加工)，既涉及应用逻辑，又涉及用户界面。

(3) 阶段，在开发过程中分阶段逐步精化和扩展应用，开发阶段作为第三维，通过不断精化需求阶段所识别的需求，产生分析模型和设计模型。

建模在数据工程、软件工程中已经广泛使用。数据工程关注应用的数据的结构方面，识别实体及其关系，最广泛的模型是E-R(实体关系)模型。软件工程关注的是程序设计所需的行为方面。如今，建模主要是基于面向对象(Object Oriented，OO)方法。面向对象建模的最重要的特性是以对象为中心的概念从结构和行为两个方面建模系统，UML几乎成为面向对象软件开发的通用建模语言，提供静态结构图(如类图、组件图、协作图、部署图)和行为图(如用例图、状态图和活动图)，也作为Web应用建模的基础。

对于Web应用建模，和传统软件开发一样，有应用域和业务逻辑，通常由用例模型、实施模型、部署模型、安全模型等一组模型来表示。但另一方面，尽管Web应用很复杂，但它总是表现为不同的Web页面的交互以及用户和Web页面的交互两种关系。因此在Web应用的建模过程中，要将业务逻辑和Web应用页面结构两个方面分别进行建模，需要不断地进行用户评估，收集用户反馈的信息。业务逻辑的建模类似于传统软件的建模，而Web页面建模则需要针对Web应用的建模方法。Web应用建模还需要支持重用、Web应用具有的特定体系结构，以及对于复杂应用，如何更好地支持个性化等方面。

4.1.1 层

要建模Web应用，必须考虑其类似文档特性的内容及其非线性的超文本导航特性，因而，把建模Web应用分为内容、超文本和展示三层。内容层即Web应用底层的信息和应用逻辑，超文本层即将内容组织成为结点以及相互之间的链接，展示层即用户界面或布局。

将内容、超文本和展示分层有助于降低复杂性。例如，在只有一份内容的基础上，可以构建不同的超文本结构以满足不同的用户组和所用设备。内容模型的目的是展示信息结构。和数据建模的数据库模式相比，这样可以消除冗余，也就是即使信息本身不断改变，信息的结构也保持不变。

要设计有效的导航，可能需要在多个结点上放置同一内容的不同超文本。将内容和超文本分开考虑，内容只建模一次，超文本结构模型只是引用相应的内容。这样的话，用户可以通过多条路径访问到信息。为了使用户在错综复杂的路径中保持清醒而不迷路，可以采用超文本建模的循环导航模式(注：超文本建模的设计模式之一)。

展示层建模的目的是提供Web页面的统一的展示结构，以使用户易于明了各个页面。需要注意的是，虽然Web应用的展现非常重要，但是我们在建模部分不强调艺术方面。

上述三层建模虽然是分开考虑，但是相互之间具有依赖关系。例如，不同的个性化超文本访问路径可以映射到同一个内容模型。我们所描述的Web应用的整体模型包括这三层，根据Web应用类型的不同，可能侧重不同。例如，给一个大数据集提供纯面向超文本的用户界面的Web应用可能需要将建模的重心放在内容和超文本结构上，而模型表示的Web应用(如在线购物门户)很可能主要是对展示进行建模。

4.1.2 方面

根据面向对象原理，为内容、超文本和展示三层分别构建结构和行为模型。结构和行为模型的相关性依赖于要实现的Web应用的类型。如果Web应用主要是提供静态信息，则需要比较少甚至不需要行为建模；而高交互性的Web应用，如提供搜索和订购等功能的电子商务应用，则需要更多的行为建模。

从映射不同层的角度出发，建议采用相同的模型表示形式展示结构和行为模型，以采用同一个CASE工具。一般来说，这种建模表示要能应对三层中每一层的特性。

4.1.3 阶段

如前所述，目前还没有一个一致且通用的建模方法以支持Web应用开发。但是不管采用哪种方法，会有分析、设计和实现等一系列步骤需要建模人员考虑。

根据Web应用类型的不同，可能需要采用信息驱动方法（如从内容建模开始），或者展示驱动方法（如从建模应用的展示开始）。Web工程中基于模型的开发方法有时候和Web项目的实践需求有些冲突，比如，建模需要更多的时间，而Web项目很短的开发周期和交付时限更需要敏捷方法。

因此，基于模型的方法和现实需求的矛盾，需要采用合适的工具加以支持，以尽可能达到自动生成Web应用。和Web应用的生存周期相对较短相比，模型能确保解决方案的持久性。另外，模型有助于团队开发人员之间，以及客户和开发人员之间更好地进行沟通。

4.1.4 适应性

如前所述，Web应用开发中，上下文信息起着非常重要的作用，它支持用户个性化、多平台交付和基于位置的服务。适应性就是Web应用针对个性化和移动计算需求，进行自适应。如根据用户偏好进行信息过滤和内容推荐，提供适应性用户界面和超媒体内容；根据移动计算中网络带宽、设备特性、时区、位置以及多渠道交付等上下文，进行自适应。适应性影响着Web应用建模的其他三维，需要在所有的开发阶段，从结构和行为两方面影响内容、超文本和展示三层。因而，将处理上下文信息看做是独立的第四维特性——适应性。

4.1.5 Web应用建模的优点

Web应用建模为Web应用的开发与维护带来了巨大的方便，具体来说，它的优点主要表现为以下几个方面。

1）更好地理解系统

开发过程中，用户需求增多，需求不确定性加剧，需求变更频繁，Web应用开发技术日新月异等原因都会增加Web应用的修改复杂度，Web应用变得越来越难以理解。而模型作为对现实的抽象，有助于Web应用的利益相关者从不同的角度理解Web应用。

2）满足协同开发项目的需要

大型的Web应用构建一般由较大型的项目开发团队来进行，团队中不同类型的人需要了解Web应用的方面不同。而通过Web应用建模，可以提供多种视图。基于这些视图，团

队对整个产品获得一致的理解。

3）满足用户需求

Web应用的价值在于它能满足用户的需要。在不清楚用户需求的情况下开发出的系统很难满足用户的需求。通过Web应用建模，有助于及时准确地捕捉、追踪用户需求，并评估需求变更对Web应用的影响。

4）开发过程可控

Web应用开发过程可控意味着项目经理清楚项目的进展，并保证代码质量，否则可能会做出不切实际的承诺，导致团队情绪低落并延误工期。因为Web应用模型已经清楚地展示了Web应用系统的蓝图，每个人都清楚他已经做了什么，正在做什么，将要做什么，从而将Web应用开发的不可控因素降到最低。

5）系统可持续性发展

传统的以项目为中心正在被以产品为中心所取代。在产品的生命周期内，用户的需求必然要发生变化。借助Web应用建模，可以从用户需求出发追踪到最后的编码实现。这样通过观察每一次的用户需求变化，可以清晰地勾勒出受到影响的代码，比如哪个类的方法要扩充，哪个包内要增加类等。

4.2 模型驱动开发

模型驱动开发（Model Driven Development，MDD）是以建立模型为主要手段的一种开发方法，以模型作为开发过程中的主要制品，模型可以自动或半自动生成元数据和实现代码。MDD不仅仅强调在开发中使用模型，也强调在整个开发过程中从系统需求规格说明到实现再到测试的所有开发阶段之间的转换。模型之间的转换提供了系统的整个生命周期中不同模型之间的自动转换能力。

采用MDD，可以使模型从代码中分离出来，建模者将注意力集中在高层概念和基本业务逻辑上，从而构建平台无关的模型（PIM），然后选择特定平台和相关代码生成工具构建平台相关模型（PSM）并进行代码生成。MDD采用一种建模语言进行建模，代码生成很大程度上通过自动化生成，如自动化生成SQL数据调用、Java或.NET特定平台代码以及Web客户端XML和JavaScript。因此，避免了大量的代码测试工作，提高了开发效率。

MDD已由不同组织以不同的架构和工具进行了实践，如RUP以MDD为基础，采用迭代的方式进行模型驱动开发，而最具代表的是OMG发布的工业界支持的MDD标准规范MDA（参见第5章）。另外，MDD工具也不断发展，支持度越来越高，如CA Gen在模型保持不变的情况下重新生成新的分布计算平台的代码，而且其业务逻辑描述语言易于使用。

敏捷模型驱动开发（Agile MDD，AMDD）和MDD的区别在于，AMDD并不强调在编写或生成代码之前创建大量的模型，而是创建一定程度上够用，甚至只有很少的模型，即敏捷建模（Agile Modeling，AM），然后就进行编码。在编码实现模型之后，或者编码过程中，再根据需要迭代进行进一步建模。

然而，对传统软件进行建模的方法不能完全满足Web应用的特性，比如UML不支持对超链接进行建模。随着Web工程的发展，过去的几年出现了多种针对Web应用的建模

方法，比如包含有导航图，以从上述层、方面和阶段三维对 Web 应用进行建模。

由于 Web 应用特性分为内容、超文本、展示和适应性，MDD 可以很好地应用于 Web 应用开发中，更好地指导 Web 应用的开发，提高 Web 应用开发的效率以及 Web 应用的质量。MDD 的另一个优点在于其灵活性，即当 Web 技术演化而引入新技术时它的灵活性。MDD 过程需要的各种演变，WebML、OO-H 和 UWE 等建模方法提供了采用 MDD 进行 Web 应用开发的很好的基础。

4.3 Web 应用建模方法与工具

Web 应用模型是 Web 应用的抽象，存在着两个主要问题：页面的特殊性和复杂性造成建模困难，建模语言规范。所以常规的建模方法一般需要解决这两个方面的问题。

Web 应用的日益发展，使得一些建模 Web 应用的方法应运而生。这些方法源于不同领域的研究，源于数据库研究领域的有 RMM、Araneus 等，源于多媒体研究领域的有 HDM、WebML 等，源于面向对象研究领域的有 OOHDM、UWE、OO-H 等，图 4.1 表示了几种典型 Web 应用建模方法的演变过程。Web 应用开发方法的共同点是便于将 Web 应用模型分为领域模型、导航模型和展示模型，经过概念建模、逻辑建模、物理建模和实现 4 个过程完成 Web 应用开发。其中领域模型、导航模型和展示模型分别描述系统的三个不同侧面，可以看成是 Web 模型的不同视图。

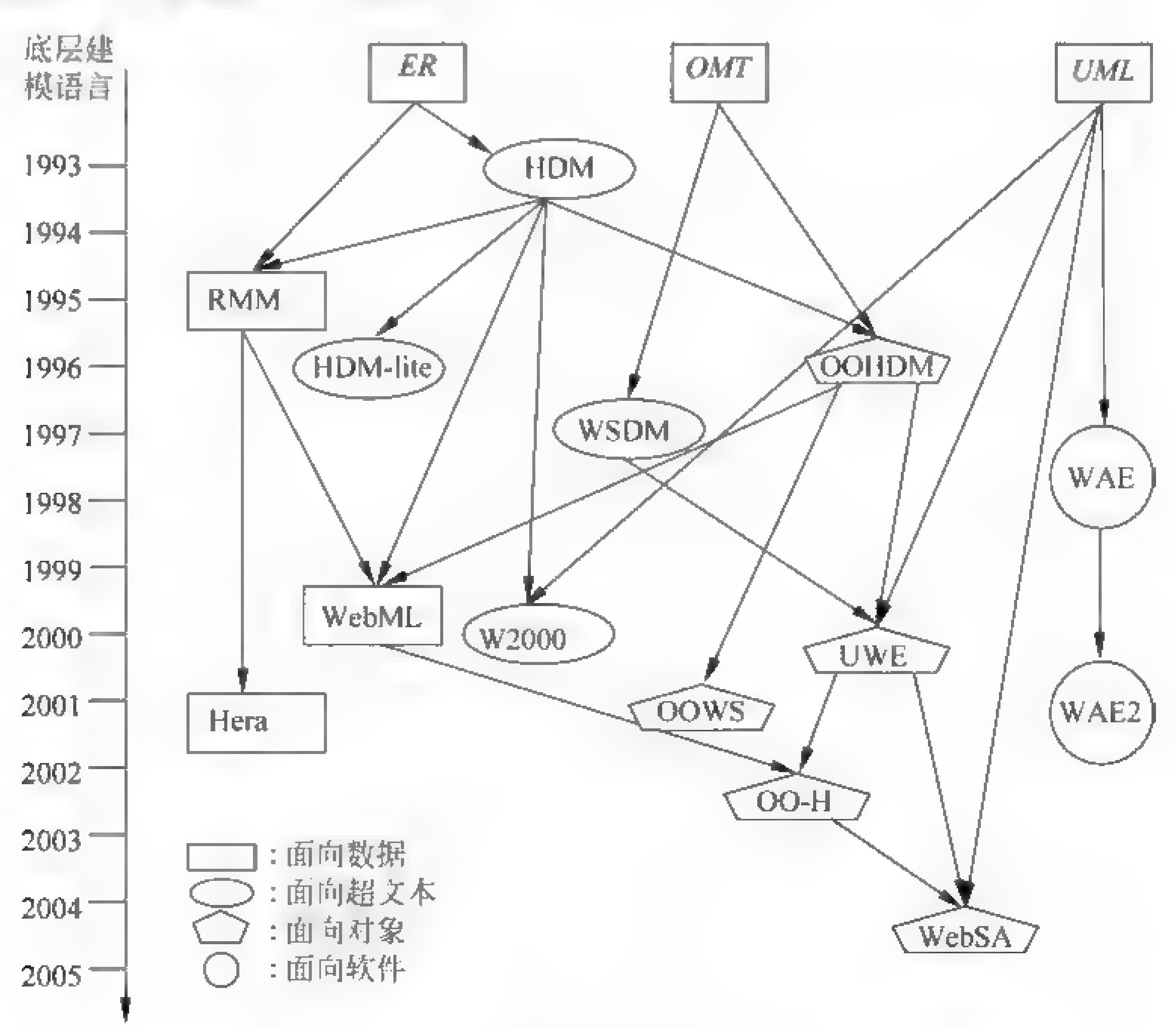


图 4.1 典型 Web 开发方法演变过程

领域模型描述 Web 应用中领域对象及其关系，是导航模型的基础；展示模型描述 Web 页面展示形式，是导航对象和导航行为的最终体现；而导航模型是 Web 模型区别于传统软

件系统模型的重要部分，描述了 Web 应用的导航特性，并起着衔接领域模型和展示模型的作用。

以上列出了 Web 应用开发方法应该具备的主要特征和功能，下面将从这几个方面考察几个典型的 Web 应用建模方法已经具备了哪些特性。

4.3.1 UWE

UWE(UML based Web Engineering，UWE，基于 UML 的 Web 工程)方法是由 Koch 于 2001 年提出，随后 Hennicker 和 Kraus 等又对其进行了完善。该方法以 UML 为基础，是一种模型驱动开发 Web 应用的系统化的方法。UWE 强调"分而治之"的概念，将内容、导航结构、业务过程以及 Web 应用的展现分别进行建模。UWE 通过定义不同类型之间的模型转换，以从 PIM 产生 PSM 和生成可运行程序，从而实现模型驱动开发过程。UWE 是一个面向对象的、迭代的建模方法，关注系统化、个性化的开发和生成 Web 应用。

UWE 主要关注 Web 应用建模，其主要特点是基于工业界标准建模语言 UML，定义 UML 元模型的扩展，并映射到 UML profile(配置文件)，其转换采用 QVT 或者 ATL 进行定义。UWE 对 UML 元类的扩展包括两个方面：①添加元属性(标记)；②添加约束，如图 4.2 所示。

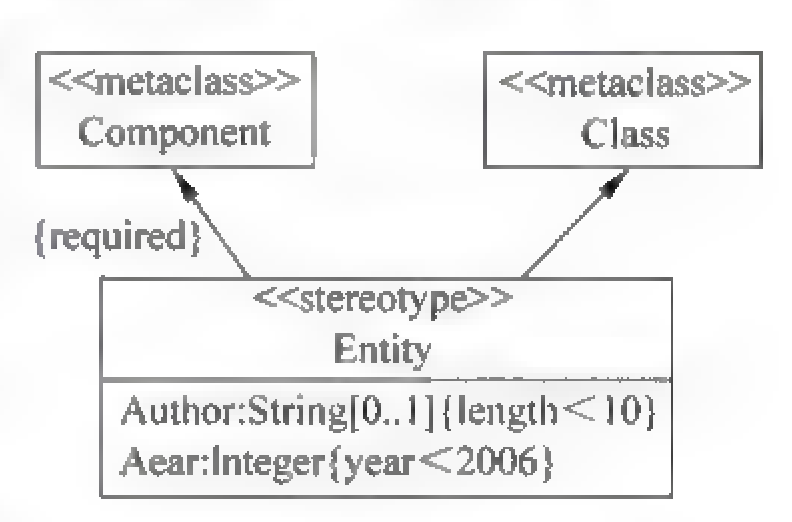


图 4.2 UWE 对 UML 针对 Web 应用的扩展

Stereotype(构造型)是 UML 提供的一种扩展方式，UML 是通用的统一建模语言，其三种核心扩展机制包括 stereotype、标记值和约束。针对不同的应用领域和实际应用需求，用户可以自定义，扩展得到更有针对性的建模语言，类似普通话和方言。OMG 的文档中提到，可以根据项目的需要对 UML 进行扩展，制定适合该项目需要的特定建模语言。stereotype 在 UML 的标准元素上增加了语义。

UWE 中主要包括需求模型、内容模型、导航模型、过程模型和展示模型。每种模型的关系描述如图 4.3 所示。

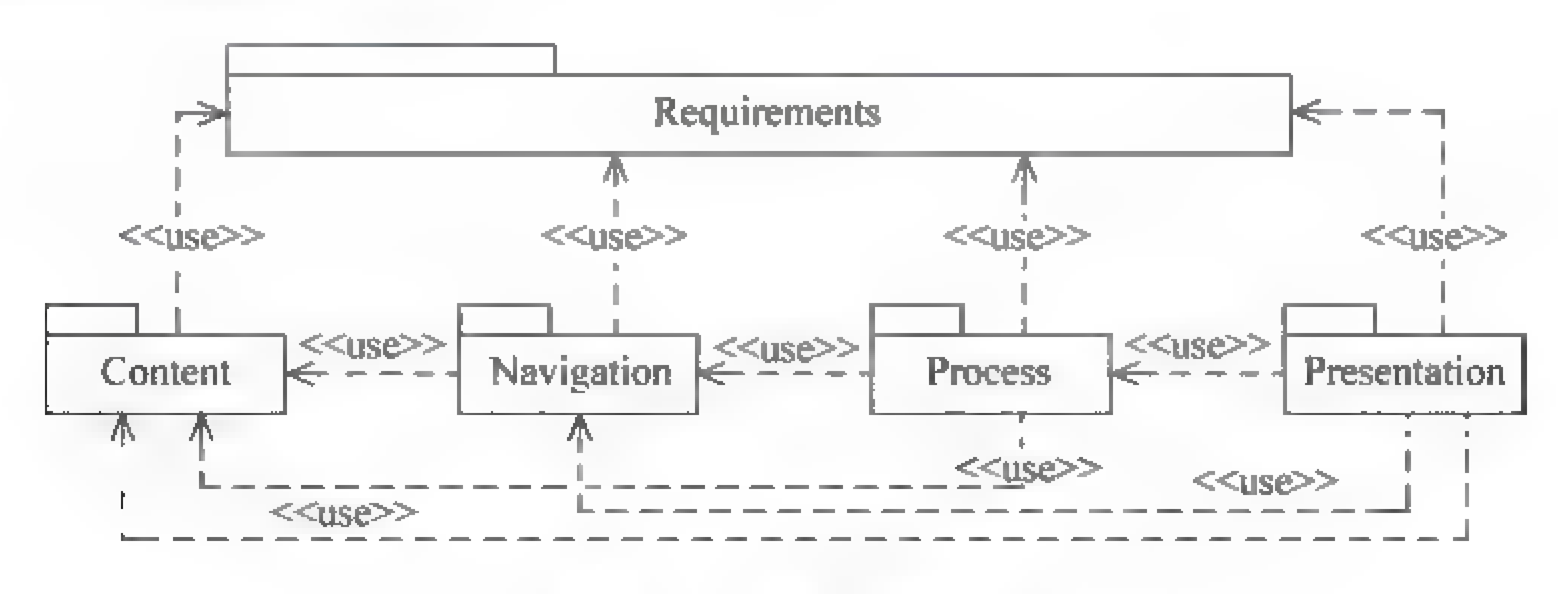


图 4.3 UWE 元模型

基于 Web 应用的功能性需求，UWE 扩展了导航链接(<< navigation link >>)、过程链接(<< process link >>)和外部链接(<< external link >>)三类链接。

1) 需求模型

UWE 设计过程是从需求模型开始。需求模型由 UML 的用例图组成，其中包含导航业

务过程用例，用<< navigation >>进行区分。例如，在新闻系统中浏览新闻视频就是一个导航用例，而添加或删除视频即为标准 UML 用例。

UML 用例图是由 UML 建模元素及元素之间的关系构成，其中 UML 建模元素主要有两个，即用例和角色；元素之间的关系有三种，即角色和用例之间的“关联”关系、用例之间的“包含(<< include >>)”与“扩展(<< extend >>)”关系、角色之间或用例之间的继承关系。

2）内容模型

UWE 中的内容模型用 UML 类图表示，提供 Web 应用和领域有关的信息的规格说明。比如，在视频库中包含的视频按分类(如娱乐、体育等)进行组织。每部视频具有名称、时间、简介、图片、类型(娱乐或体育等)等。将视频库设计为 VideoCollection 类，视频设计为 Video 类，那么 Video 和 VideoCollection 类中包含向视频库中添加视频或从视频库中删除视频的方法。

类图是静态视图。与 OOHDM 方法中的类图一样，UWE 方法中的类图尽量忽略应用程序的导航、展示和交互等方面的内容，仅仅表达应用程序的概念框架。

3）导航模型

在需求模型和内容模型的基础上，导航模型是用于展现 Web 应用系统的超文本结构，用结点和链接进行表示。对于具有导航性的类，用扩展的 stereotype << navigation class >> 来表示信息获取，如 VideoCollection 或 Video；用<< process class >>定义发生事务处理的导航结点，如 AddVideo 和 RemoveVideo。用关联关系建模直接链接，尤其是关联关系<< process link >>的一头是过程类。有时需要一些专门的导航结点，以便于组织链接。比如一个导航类的一些实例通过<< index >>类表示，而一些可选的链接用<< menu >>类来表示。

在导航模型中，每个过程类可以通过过程结构(类图)进一步定义过程中用到的其他类，并通过过程流(UML 活动图)建模过程的数据和控制流。例如，在 Web 应用中可以把用户交互建模为 UML 的动作，用 stereotype << userAction >>。

可以将导航模型分为两种类型：导航结构模型和导航访问模型。前者描述超文本结构，即内容模型中包含的类和对象映射为超文本中结点(页面或文档)以及这些结点之间的链接。超文本结构模型通常被视为内容模型的上层视图。导航模型元素是导航类和直接导航关联，从内容模型中的类以及关联映射过来。导航结构模型定义导航对象是怎样被访问的，借助一些模型元素来描述导航结构，这些元素包括菜单、索引、向导、查询、外部结点和导航上下文。后者是前者的精化模型，描述哪些结点可以通过导航方式来访问，以及如何通过导航访问结点。借助于 UML 的 Stereotype 可以从内容模型导出导航访问模型，导出时去掉用户不必看见的内容类，同时调整类之间的关联关系(调整的主要内容是尽量缩短类之间的路径，以免用户在浏览时过多地翻页)。

4）展示模型

展示模型是 Web 应用用户界面(UI)设计的结果，提供 UI 的抽象视图，详细描述了用户可见类对象和存取结构(如索引、向导、菜单、查询)在什么位置出现，以什么面貌出现。结构类构造为<< presentation group >>模型，表示每个导航类，其中包含的 UI 元素(如文本、图片和按钮等)构造为<< text >>、<< image >>和<< button >>等，来表示所使用的窗口小部件。如果展示类没有包含在其他展示类中，则表示它是 Web 应用的顶层页面。另外，一个展示类是为每个用户动作而定义的。因此，展示模型中有很多展示类。

5）适应性模型

UWE 采用面向方面建模（Aspect Oriented Modeling，AOM）技术进行适应性建模。AOM 一方面可以使得系统功能和个性化方面系统地进行分离；另一方面可以减少冗余。UWE 通过使用 stereotype UML 包来支持 pointcut 和 advice 两个部分。一个方面是一个语句（可以是图形化的），表明除了在基本模型中指定的特性之外，包中的每个横切点模型元素也具有 advice 指定的特性。也就是说，完整的约束既包括一般系统功能，又包括模型和方面的横切（cross-cutting）特性，也称为 weaving。

6）模型一致性

UWE 中不同模型之间具有一致性约束或生成关系。导航模型的基础是底层内容模型，超文本模型或多或少和内容模型密切相关。一方面和类型层次密切相关；另一方面和实例层次密切相关。和导航模型与内容的映射关系类似，UWE 的展示模型和导航模型之间也可以进行映射，通常认为一个结点的所有实例将在展示层展现，并且需要考虑 Web 应用中用户的交互行为。适应性模型影响前面所有的 Web 应用建模过程，变化可能会在一层或者也会影响几层。因此，将适应性模型根据内容模型、导航模型和展示模型分别进行处理。

由于 UWE 采用 UML 作为统一的模型符号基础，结合 UML 和扩展，模型构建相对成熟（基于 UML 的用例图、类图、顺序图、状态图、活动图、部署图等），模型之间的映射关系相对明确，在 UWE 中主要从建模角度提出了映射原则。模型的集成和连通能力得到了加强。

尽管 UWE 方法不考虑 Web 应用的实现，但实际上可以利用与 UWE 方法相关的 CASE 工具如 ArgoUWE，将上述的模型图转换为应用程序框架。开发人员在生成的应用程序框架的基础上，可以快速地完成整个应用程序的设计。本章后续建模将主要采用 UWE，进一步描述该建模方法在 Web 应用建模时的使用细节。

目前，支持 UWE 的主要工具有 ArgoUWE（ArgoUML）、UWEet、MagicUWE 以及 UWE4JSP。

UWEet 是开源 UML 工具 UMLet 的扩展，UMLet 支持绘制 UML 图，并导出成 eps、pdf、jpg、svg 和剪贴板，与 Eclipse 共享图形。UWEet 提供了包含 UWE 的 stereotype 的画板。MagicUWE 是 MagicDraw16.8 的插件，用于在 MagicDraw 中使用 UWE 进行 Web 应用的设计。图 4.4 所示为 MagicUWE 作为 MagicDraw UML 16.8 插件的界面示意图（本章第 4 节起的建模采用本工具）。而 UWE4JSP 是 Eclipse 的插件，支持 UWE 的所有模型，用于自动生成 JSP 平台上的 Web 应用。这几款工具都充分利用所基于平台的所有功能，扩展了 UWE 的支持。有关 MagicDraw 和 Eclipse 的进一步信息，参阅相关资料。

ArgoUWE 是一个基于开源工具 ArgoUML 的针对 UWE 的扩展，它使用 ArgoUML 的常用图形化用户接口，支持 UWE 中的新模型，同时还支持半自动生成这些模型及其一致性检查。在 ArgoUME 的图示中，使用者可以像使用 ArgoUML 一样，添加、删除、复制、粘贴模型元素，也可以编辑属性。ArgoUWE 继承了 ArgoUML 项目浏览空间的四个框架：导航面板、多编辑器面板、评价面板和细节模板。ArgoUWE 目前已经不再继续维护。

UWE 对于 RIA 建模能力有限，比如 Google Map。因此，基于 UWE 针对 RIA 进行轻量级的扩展，形成 UWE R。UWE R 保留了 UWE 的所有特性，扩展出针对 RIA 的导航、

展示和服务器端交互(处理过程),如导航扩展 stereotype ≪RichNavigationClass≫、≪RichNavigationLink≫、≪clientProcessClass≫,展示扩展≪Canvas≫、≪Panel≫,以及处理过程扩展≪ControlMessage≫,等等。每种扩展都看做是表达 RIA 概念的插件。

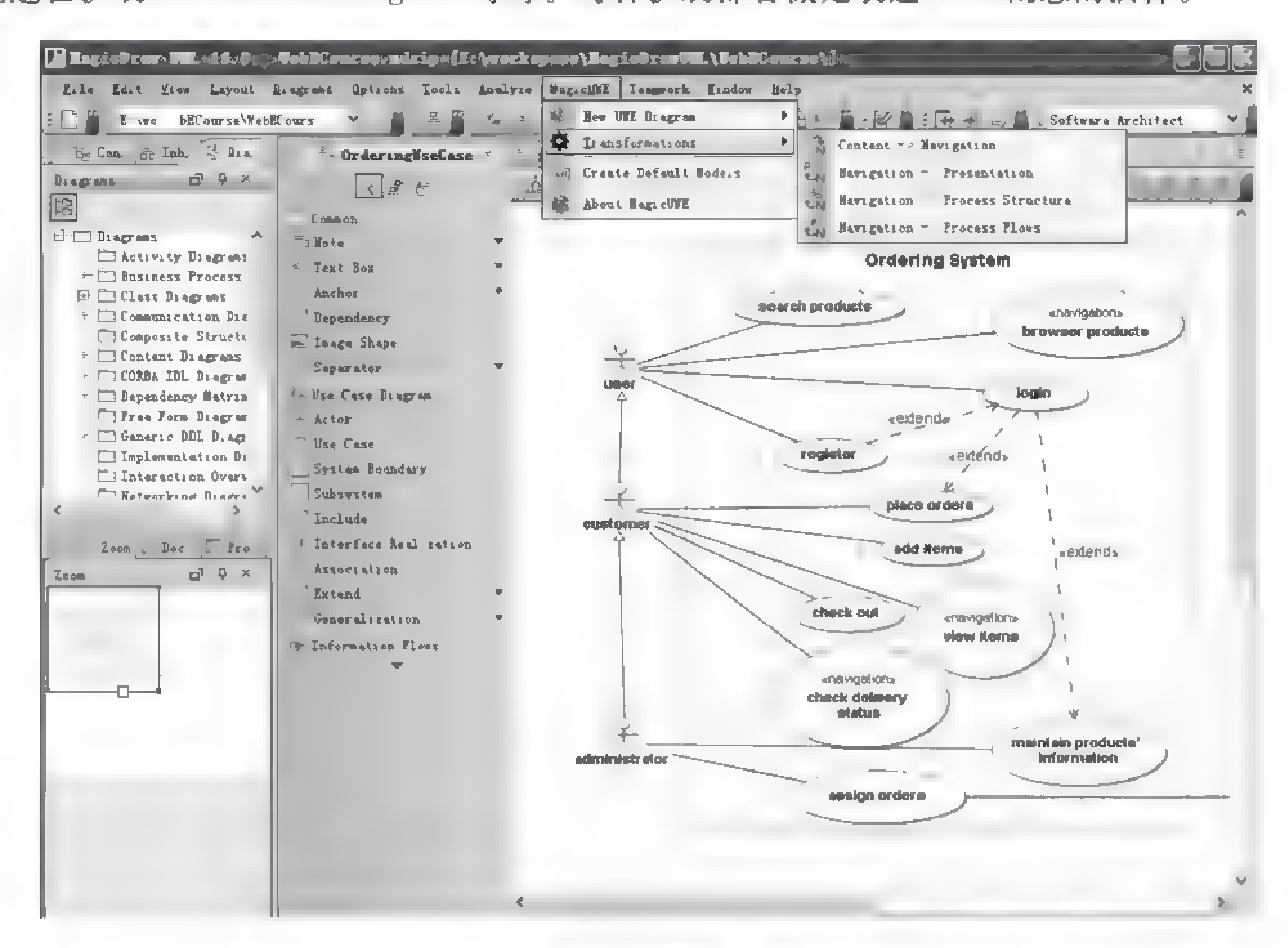


图 4.4 MagicUWE

4.3.2 WebML

WebML(Web Modeling Language,Web 建模语言)是 W3I3(Web-based Intelligent Information Infrastructure)项目定义的 Web 模型描述语言,通过图形符号和 XML 语法进行描述。它是 HDM lite 的进一步演化,是在概念层次上描述复杂 Web 应用的符号体系,也被称为"针对 Web 的 UML"。它是比 RMM 出现得更晚、表示更加丰富,它对工作流建模、展示和内容的适应性、个性化、设计模式、多层访问具有强有力的支持。WebML 在 Web 应用开发过程中保证了模型驱动开发,其规范独立于客户端和服务器端。

WebML 方法能使系统设计人员抛开具体细节,而在一个较高的层次上描述站点的核心特征。借助于 CASE 工具(如 WebRatio),WebML 方法中的每一个概念都可以用比 UML 更直观的图形符号来表示,使得开发团队中的非技术人员也能理解。WebML 方法完全支持 XML 语法,每一步设计的结果既有直观的模型图,又有对应的 XML 文档,这一特点使得设计阶段的工作完成后,Web 应用软件代码的自动生成变为可能。

WebML 的主要目标有以下 6 个方面的内容。

(1) 运用高层的描述表示 Web 应用的结构。

(2) 对同一内容建立多个视图。

(3) 分离信息内容。将页面、导航呈现从它们的合成中分离出来，单独定义和演进或求精。

(4) 将设计过程中收集到的媒体信息保存到数据存储中，在Web应用生命周期中这些信息将被用于动态产生Web页面。

(5) 为支持个性化策略和一对一的Web应用，可明确地建立用户或用户组模型。

(6) 能规范地描述与Web应用内容修改相关的数据维护操作以及任意的与外部服务的交互。

WebML能够在高层次上清楚地划分为如下模型分层描述Web应用。

1) 结构模型

结构模型(Structure Model)用于描述Web应用的数据内容，它通过实体和实体之间的关系来描述内容。为描述冗余和计算信息，结构模型还定义了一种简单的类似于OQL的查询语言，用于表达导出信息。目前，WebML还没有计划建议为结构模型提供新的符号表示，而是和传统的实体关系(E-R)模型、ODMG面向对象模型以及类图兼容。在结构模型中，数据用实体表示，实体可以具有相应的属性，各实体间通过关系联系。

在结构模型中，实体是基础，它与生活中的实际对象相关。实体的图形化表示方法是一个分成上下两部分的矩形框，上半部分标明实体的名称，下半部分列举该实体具有的属性。图4.5展示的是一个简单的结构模型，Artist、Album、Review和Track为4个实体，每个实体具有两个属性。关系是展示结构模型中实体之间相互连接的语义关系，与UML中的关联类似，WebML中关系同样可以使用多重修饰，表示实体之间的数量对应关系，如图4.5所示。

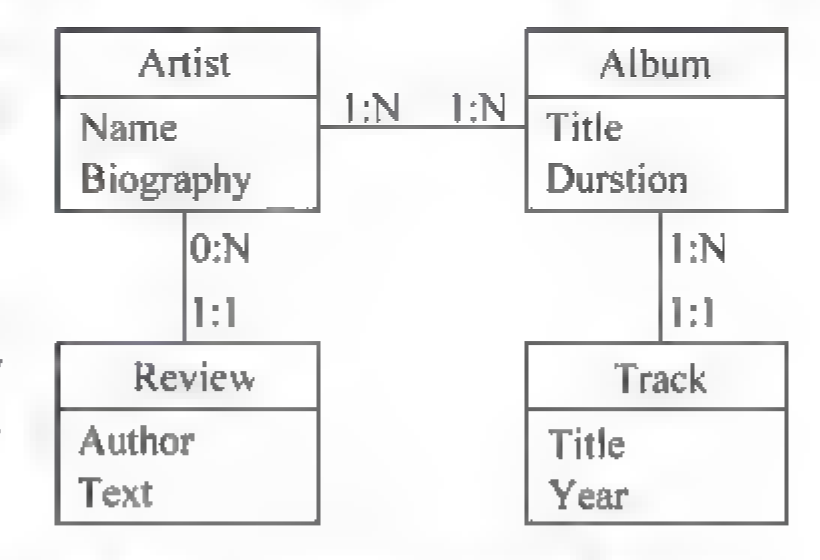


图4.5　WebML中结构模型示例

实体Artist和实体Album之间的直线表示它们之间的关系。1∶N表示的是一名艺术家可以有多部影集。

数据是WebML描述Web应用的基础，结构模型是WebML模型的基础，实体在WebML中非常重要，是WebML很多元素的基础。采用WebML建模，Web应用首先要设计数据模型，否则其他设计无法开始。

2) 超文本模型

超文本模型描述Web应用中的超文本，每个不同的超文本定义一种Web应用视图，Web应用视图描述由两种子模型组成：组成模型和导航模型。

组成模型(Composition Model)描述组成超文本的页面，以及页面是由哪些内容单元组成。组成页面的单元有数据、数据集(Multi-data)、索引、过滤器、滚动和链接6类单元。其中数据单元用于单个对象信息的发布，数据集、索引、过滤器和滚动单元表示浏览一组对象，因此也被称为容器单元。组成单元定义在Web应用结构模式之上，设计者确定每个单元底层的实体或关系。

导航模型(Navigation Model)是Web页面间的链接关系拓扑模式，它表示Web页面和内容单元如何链接而形成超文本结构。WebML的导航模型是在组成模型的基础上定义链接，体现导航的链接有上下文无关(Hyperlink)和上下文相关(Infolink)两类。上下文相关链接是指链接的单元在应用的结构模式中存在语义上的关系。导航链接带有从源到目标单

元的上下文信息。上下文用于确定在目标单元中显示的实际对象。上下文无关链接表示完全随意拦截页面，和所包含这些单元的结构概念无关。

在实际的应用中，导航链接经常以导航链的形式出现，WebML还分析总结了多步索引、筛选索引、索引向导、环等Web导航模式来体现这些特征。

3）展示模型

展示模型（Presentation Model）是Web页面的物理外观和感觉，如Web页面的布局和图形外观，由抽象XML语法实现与具体的输出设备和页面呈现的具体语言无关性。WebML的页面展示模型或是页面特定的或是通用的。前者说明一个特定的页面的展示模型中包含对页面内容的显式引用，而后者表示页面的布局是预先定义的模型，和特定页面内容无关。

4）个性化模型

个性化模型（Personalization Model）是个性化适应性数据的记录模式。用户对系统的个性化要求越来越高，WebML区分两类用户即预定义实体：用户（User）和组（Group），并显式地在结构模式中进行个性化建模。利用这一特性，可以存储一些与组或者用户个人相关的内容，比如购物建议、最喜欢的内容、图形定制的资源等。然后，类似OQL的声明表达式也可以加入结构模式，可以根据User和Group产生User和Group中存储的信息内容。定制的内容可以用于组成单元或定义展示规范。另外，高层业务规则采用简单XML语法编写，可以用于定义应用相关的事件，如用户点击和内容更新。业务规则通常会产生新的相关信息（如购物历史）或更新内容（如添加新的和用户喜欢批评的优惠推荐）。查询和业务规则提供两种可选泛型（如声明泛型和过程泛型），以便高效地表达和管理个性化需求。

WebML的理论比较简单，容易学习，它非常适合规模比较小的公司在有限的财力和物力的条件下对员工进行培训，员工能够快速上手设计开发小型Web应用。WebML的元素比较少，对Web应用做了最简单的抽象，适合从较高的抽象层次对Web应用程序建模。另外，WebML适合数据密集型的Web应用建模。这类Web应用以大量的数据为基础，服务器从数据库中抽取数据并通过浏览器在Web页中展现，只有简单的数据查询、更新、删除操作，而没有非常复杂的业务逻辑，正好与WebML的特点相吻合。

但是，WebML也存在不足之处，WebML的内容单元都依赖于一个实体，而现实的情况是，很多时候一个实体不足以表达Web应用需要展现的内容，不能完全满足Web应用对数据展现建模的要求。Web页面有一个重要的元素表单，它包含了很多元素，如复选框、按钮和下拉列表等。它们都是用户与服务器通信的重要手段，而WebML只对表单的输入框元素有相应的元素描述。WebML最大的不足之处是其模型主要是静态模型，缺乏对复杂功能特别是动态Web应用及后台业务逻辑处理过程的建模能力。主要原因是WebML是在超文本背景下发展起来的建模方法，主要着眼于信息的组织、展现和导航，而Web应用的功能（特别是核心业务逻辑）往往被忽略。

支持WebML的建模工具有WebRatio，它允许数据模型和Web应用视图的视觉规范，并且能够自动生成J2EE代码，它包含有一组Eclipse插件，并且能够很好地利用这一集成开发环境的所有功能，它也支持模型和代码生成、模型检测、项目文档等个性化的扩展。

WebRatio的主要特色在于它对Web应用以及Web服务开发提供了一组完整的模型驱动工程方法，它可以直接将模型转化为可运行代码，而且通过Eclipse上的接口联合所有

的设计和开发活动，其中包含模型的可视化编辑、展示方面的定义、集成开发环境新增组件的扩展以及代码生成规则。同时，它将所有的设计制品存储到一个普通区域（Eclipse 工作空间），并且使用版本控制工具与合作工作系统管理它们。最后，WebRatio 具有很强的扩展功能，支持任何存在的 Eclipse 插件。

4.3.3 HDM-lite

HDM（Hypertext Design Model，超文本设计模型）提出了不同设计“维”的 Web 应用建模概念，即内容、导航/交互和展示，被后续众多超媒体建模方法采用。1993 年 Garzotto 提出 HDM，它是建立在 E2R 模型基础上，包含结构化链接（Structural Link）、透视图链接（Perspective Link）和应用链接（Application Link）。

1998 年 Fraternali 和 Paolini 扩展了 HDM，发展出 HDM lite，强调自动化的开发过程和 Web 应用的自动生成。主要是定义结构、导航和展示，分别用 HyperBase、Access、展示模型来表达。

HyperBase 是结构模式（Structure Schema）的实例，是一个扩展的 E-R 图，由实体和链接组成，即实体、组件（实体子结构）、有类型的属性、实体和部件间的语义链接、数据约束等。结构模式描述组成 Web 应用的基本对象的结构属性。而导航模式指明从一个对象到另一个对象具有的行为，即 Traversal Schema（遍历模式，描述从源到目标的直接导航）和 Web 应用中可以访问到对象的访问路径，即 Access Schema（访问模式）；另外还定义了 Collections（描述集合，可以嵌套定义，共分为 8 种子类型），导航模型（Index，Guided Tour，Indexed Guided Tour，ShowAll）。同一个结构模式可能有多种不同的导航模式来表示不同的访问相同信息的路径。

HDM-lite 的展示模型是一系列样式单的集合，使用类 SGML 的语法来表示。多个展示模型可以被映射到一个结构/导航模型但没有提供设计结构。展示模型的基本单元是页面。一个样式单表针对一种特殊类型的页面，从逻辑上看可分为两层：布局层，每个页面建模为一个 Grid，Grid 中的每个单元包含了展示元素；元素层，有两种类型的元素，内置的和用户定义的，所谓内置元素是前面所涉及概念的抽象表示，用户定义的则为图形设计人员设计的 Banners，Applets 等。

HDM-lite 的主要特点是为了自动生成的目的对 HDM 做了适当扩充，完成从概念模型到逻辑模型乃至物理模型的转换。这些转换由 Auto Web 系统工具支持，自动产生程序数据模型、导航模型和展示模型。这些逻辑模型进一步被用来自动产生页面。但此自动生成主要是针对静态只读信息页面。

4.3.4 OOHDM

OOHDM（Object-Oriented Hypermedia Design Method，面向对象超媒体设计方法）是 Gustavo Rossi 等人开发的 Web 应用开发方法。OOHDM 采用面向对象框架中抽象和组合机制，既可以简洁描述复杂信息，又可以描述复杂导航模式和界面转换规范。

OOHDM 把导航设计作为重要的一个阶段分离出来并建立相应的模型。导航模型又分为导航类模型（Navigational Class Model）和导航上下文模型（Navigational Context

Model)。导航类模型是领域模型在导航设计中映射的视图，是通过面向对象定义语言对领域对象属性进行组合或剪辑，以及进行属性和关系筛选而建立的。模型元素为结点和链接，描述了导航对象及导航对象之间的导航关系。导航上下文模型中利用6种上下文元素描述导航空间结构，上下文是可以嵌套定义的。导航上下文模型中还有进入结构(如向导、简单索引、动态索引等)来描述进入上下文和导航对象的结构。

OOHDM将Web应用的开发周期分为如下阶段以支持迭代开发或原型过程模型，每个阶段关注特定的设计内容，并构建面向对象模型，其中应用了分类、聚合和继承等面向对象概念来提高抽象和重用能力。

1) 捕获需求

如第3章所述，捕获需求主要是捕获利益相关者对Web应用的需求。首先是识别参与者及其所要完成的任务，然后收集情景，并用用例图进行建模，即表达为用户交互图(UID)，这些图提供了用户和系统之间的交互的图形化表示。UID由参与者进行验证，并构建一组UID映射为概念模型的原则。

2) 概念设计

使用面向对象的建模原则和一些扩充的原语(如多值属性)，建立Web应用所涉及应用领域的概念模型(如对象类图)。概念模型包括子系统、类、类之间的关系。概念模型构建时，并不考虑用户和任务的类型，只关心应用领域的语义。

3) 导航设计

根据导航背景描述超媒体应用的导航结构，包括各种类型的导航类，如结点、链接、索引(Index)、向导(Guided Tour)等，其中链接从概念模型的关系导出，相同的概念模型可能导出不同的导航模型，导航模型可以被看成是概念模型上的视图。导航设计时考虑潜在用户及其任务的类型。结点表示逻辑窗口(或视图)。

4) 抽象界面设计

通过根据界面类定义的“感知对象”来构造抽象界面模型。界面类由一些基本类型(如文本字段、按钮等)和其他界面类递归聚合而成，描述可感知对象，描述导航对象的界面展示，定义界面布局。通过如何处理外部的和用户产生的事件以及界面和导航对象的交互和界面的转换来描述界面的行为。抽象界面设计采用抽象数据视图(Abstract Data View, ADV)、配置图、ADV-图表(ADV-Charts)和设计模式进行展示，和导航模型相映射。

经过上述4步以及不断迭代，最后基于某种体系结构，将界面对象映射为目标环境的实施对象。

OOHDM主要采用类图来描述静态页面的导航结构。至于导航的复杂行为语义，可以通过使用一个面向对象的状态转移模型Navigation Charts来描述。它用ADV来描述导航对象和其他界面对象如菜单条、按钮等的结构布局。界面复杂行为方面则采用ADV图表来描述。

OOHDM的优点众多，主要表现在以下几个方面：①使用了面向对象的分析方法；②使用三个层次分离关注点；③通过引入导航上下文的概念可以定义出简明的导航模型；④定义了一个合理的开发过程。但同时它也存在着诸多不足之处：①在导航模型和展示模型的描述中使用了非规范的描述方法；②映射关系不够清晰，且非自动化。

OOHDM Web是一个采用OOHDM描述的设计工具，但主要针对的是只读性的Web

应用。软件自动生成由 HTML 代码和 OOHDM Web 函数库调用相结合的 Web 页面，所以它产生的页面必须在 OOHDM Web 环境下运行。

4.3.5 WebSA

WebSA(Web Software Architecture)是另一种 Web 领域中基于 MDA 泛型的模型驱动方法。WebSA 基于 UML、UWE 以及 OO-H 方法，试图通过引入软件体系结构(架构)，使系统的功能需求和非功能需求都得到较好的满足，即建模 Web 应用架构。它使用 MDA 标准来规范化描述，这使得应用 WebSA 开发 Web 应用的体系结构具备以下三个重要的特性。

(1) 提高了 Web 应用的开发速度。

(2) 使得 Web 应用接口与已存在模块的集成更加容易。

(3) 减少了 Web 应用的开发成本。

WebSA 和 MDA 类似，强调构建平台无关模型，然后自动构建平台相关模型，平台相关模型作为生成可执行代码的基础。

WebSA 方法提出使用 UML 模型表达 Web 应用的开发过程，以及 QVT 转换，如图 4.6 所示，第一步转换 T1 是集成 Web 应用架构模型与结构和行为模型。在 WebSA 中将这些结构和行为模型称为功能模型，其关注的是 Web 应用的功能需求，而架构模型主要是基于非功能需求。功能和架构的合并结果是集成模型，其中包含配置和子系统视图。下一步是从平台无关模型到平台相关模型的转换(T2)，转换时包含平台相关模型，比如 Java EE 或 .NET 等。

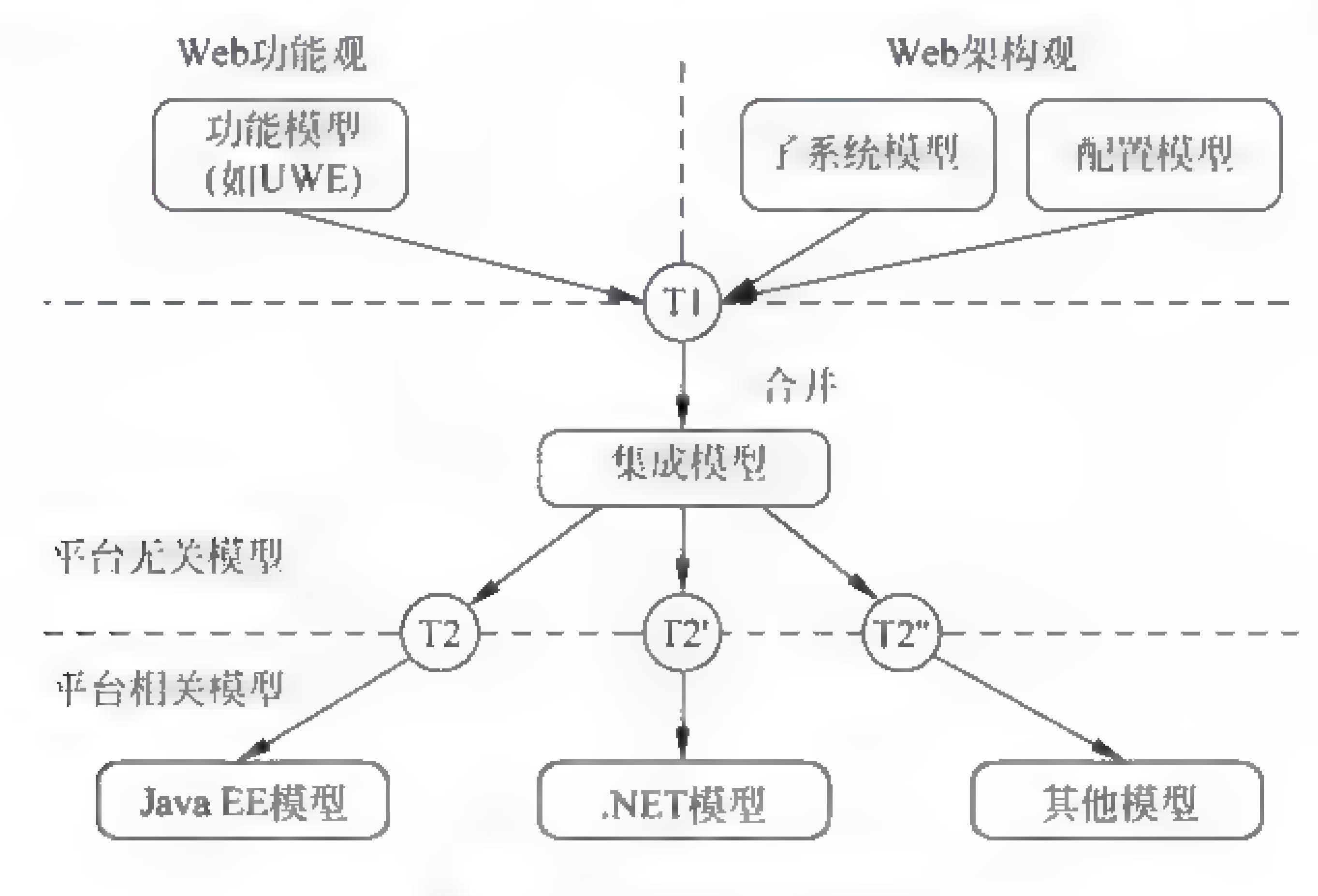


图 4.6 WebSA 开发过程

4.3.6 其他方法

Web 应用开发方法还有很多，比如，RMM、WebComposition、OntoWebber、W2000、OOWS 和 WAE2 等。

1. RMM

RMM(Relationship Management Methodology，关系管理方法论)是一种以 E-R 模型为基础的早期的建模方法，是一种用于设计、构建和维护 Internet 及 Intranet 上 Web 应用

的方法。它的根本目标是降低动态数据库驱动的Web应用的维护成本,提倡系统进行形象化表示,以便展开设计上的讨论。它是一个渐进式的过程,包括Web页可视元素的分解,以及这些元素与数据库实体的关联关系。RMM是一种用于动态Web应用创建和维护的方法。

RMM中模型分为三层:内容层、超文本层和展示层。其中内容层单独建模,采用E-R模型,通过识别内容(数据)对象、属性、关系和组成Web应用信息空间的各种类型指标来定义应用的信息域,并最终在超文本结构中变成结点和链接;而展示层结合超文本层进行定义,Slice Design(切片设计)用于展示层实体属性分组,切片是展示单元,用作超文本的页面展示;导航设计用于确定各种切片之间的链接,导航设计通过对来自E-R图的链接标签的所有切片进行选择来对这些页面进行连接。RMM用访问元来说明导航,如链接、分组(菜单)、索引和向导等,用于设计详细的原型或Web页面。

RMM定义了渐进的过程对模型不断精化,常包含如下一组活动:可行性分析、实体关系(E-R)建模、切片设计、导航设计和构建。每个活动都由一系列步骤和一组工作产品组成,例如,一个"可行性文档"是在可行性分析结束时产生的产品。RMM有三个活动和信息设计的讨论有关:E-R建模、切片设计和导航设计。

支持RMM的工具有RMCase,由J. Nanard和M. Nanard联合研制。实际上,RMCase不仅仅是一个CASE工具,还是一个完整的WIS(Web Information System,Web信息系统)开发环境。设计人员既可以利用该工具进行建模设计,还可以在此基础上生成HTML代码。

2. WebComposition

在WebComposition方法中,Web页面、链接等实体都被建模成构件。构件可以建模任意粒度的Web实体,例如,一个页面可以是一个构件;页面中的一张图片也可以是一个构件;构件可以通过聚集(整体-部分关系)或泛化(继承关系)关系与其他构件关联。构件包含状态和行为两个部分,状态是一些属性(名-值对)的集合,行为由一些操作构成。构件之间还能以原型继承的方式互相重用。WebComposition系统支持在软件开发生命周期内,以较细的粒度对Web应用进行建模,支持Web应用构件模型的持久性存储。构件模型可以递增地映射到基于文件的Web资源之上,而不影响已经存在的Web应用。

WebComposition支持显式的重用,利用一致的对象模型实现了设计模型到实现的平滑过渡。但是WebComposition建模的大部分内容是静态的HTML和Web应用展示层中的内容,而且,很多Web应用的实现代码都需要由用户在WebComposition指定的地方写出。因此,WebComposition方法是面向实现级别的方法,此方法经常用在OOHDM的实现阶段,使OOHDM的设计模型能够平滑地向实现模型过渡。

3. OntoWebber

OntoWebber利用本体作为构造Web应用模型的基础。OntoWebber提出6种不同类型的模型,描述Web应用的不同方面,它们分别是:领域模型(Domain Model)、个性化模型(Personalization Model)、维护模型(Maintenance Model)、导航模型(Navigation Model)、内容模型(Content Model)和展示模型(Presentation Model)。前三种模型是针对特定站点

(Site-Specific)的，一个 Web 应用一般只有一个此类型的模型，后三种模型是针对特定网站视图(SiteView-Specific)的，一个 Web 应用可以有多种此种类型的模型。

OntoWebber 利用 Daml+Oil 作为本体的描述语言分别形式化地描述了上述的 6 种模型。其模型和思路基本上借鉴了 WebML 提出的模型和思路，并在 WebML 的基础上引入本体，方便了 Web 应用之间的集成。OntoWebber 不仅利用本体建模领域内的概念，而且利用本体表示 Web 应用自身的模型，例如，导航模型就有相应的导航模型本体，方便了不同 Web 应用模型之间的集成。

4. W2000

W2000 是另一个 HDM 的扩展，是一个 Web 应用的框架，基于已经存在的两个技术：UML 以及 HDM。UML 和 HDM 的集成使模型的集成成为可能。但 W2000 在展示层设计方面比较缺乏。

5. OOWS

OOWS(Object Oriented Web Solution，面向对象 Web 解决方案)是 OO 方法的延伸，是一个模型驱动的 Web 应用构建方法。它强调 Web 应用从需求获取到模型构建，经过模型转换到最终实现的开发全过程，它推动 Web 应用规范化。OOWS 开发的两个主要步骤是获取 Web 工程的概念模型和开发方法，其中概念模型主要用来获取详细的系统需求。

6. Araneus(ADM)

Araneus 是从数据库研究领域发展而来的，更强调内容层和多媒体层的设计。其中内容层采用 E-R 模型对领域对象进行建模，多媒体层把多媒体的设计分成概念设计和逻辑设计两个阶段。分别用 NCM(Navigation Conceptual Model，从 RMM 方法中发展而来，主要在概念设计阶段完成)和 Araneus Data Model(ADM)形式化。ADM 的基础特性是 Page-Scheme 符号，允许基于页面类型和链接描述 Web 应用，而且一旦 ADM 方案建立后，一种特殊的语言 Penelope 就可以从数据库结构映射到超文本结构，并自动生成 HTML 页面。

为解决商业过程建模问题，新版 Araneus2 提供了一种机制使得构成商业过程的活动可以组织到一起。

7. OO-H

OO-H(Object-Oriented Hypermedia)是一种面向对象的用来构建复杂 Web 应用的工程方法，所采用的形式模型是扩展的 UML。它基于 XML，和其他方法的主要区别在于能应对复杂功能需求。主要强调 Web 页面和已有业务模型间的集成，提供了调用服务的机制。

在领域模型基础上，OO-H 增加了导航视图和展示视图。用来捕获 Web 应用的静态和动态特性。其中导航模型是通过为每一类用户建立不同的 NAD(Navigation Access Diagram，导航访问图)来实现的。每个 NAD 对应着相关用户导航需求的信息、服务和导航路径。当 NAD 构造后就可以根据映射规则生成默认的用户界面。

NAD 的构建基于 4 种类型的模型元素：导航类、导航目标、导航链接和汇集。导航

类是导航对象的描述，根据用户的访问权限和导航需求对领域类属性和方法加以扩展而得到；导航目标是一个模型元素的集合，用来描述用户的某项导航需求，在导航目标中一般要定义入口点；导航链接定义了导航关系；汇集是在导航类或导航目标上定义的层次结构。

在OO H中，设计过程是在领域模型基础上建立导航类视图，进一步用导航目标汇集组织导航空间。而不同类型的导航链接则定义了导航关系，这些建模元素的采用可以定义复杂的静态页面导航结构，而对动态页面和导航的行为则没有描述。在导航模式的支持方面，OO H提供了丰富的模式分类，其中导航模式分为静态模式和动态模式，动态模式中又分为流控制模式和跳转模式等，可以采用不同粒度的模式来简化设计和加强重用。在OO H方法中还定义了导航视图和展示视图之间的映射规则，可以最终产生用户界面代码，但生成的页面是简单的静态页面，实用程度仍然不够。

至于商业过程的处理，OO H采用和UWE类似的思想。由于采用了面向对象的建模思想，也体现出一定的模型集成能力。

8. WSDM

WSDM(Web Services Distributed Management，Web服务分布式管理)是OASIS(Organization for the Advancement of Structured Information Standards，结构化信息标准促进组织)批准的标准，是以用户为中心的方法，着眼于系统的定制。它试图解决信息过时、链接丢失以及允许不同用户浏览不同信息等可用性问题。其主要思想是先分组用户，根据不同用户特征建立"观察"，然后才开始概念设计。

WSDM允许来自不同开发商的管理软件更容易地进行互操作，能够进行端到端甚至是交叉式的企业管理。WSDM提供了在信息技术环境下对资源的特性进行识别、检查以及修改的标准，管理应用程序可以利用它们的传递功能，并且可以增大管理软件能够取址的资源数目和类型，一段时间过后，这将会减少该类应用程序的成本，并会扩大其潜在功能。WSDM提供了能够以标准的方式使用Web服务显示管理接口的能力，任何管理开发商都可以使用这些Web服务接口的能力，而不管内部的测试设备如何操作，这样就减少了许多所需的自定义支持。

WSDM工作组管理体现在各种设备、操作和事件中，它们共同提供了完成特殊管理任务的信息。WSDM定义了一个可扩充的基本标准性能：身份、描述、规格、配置、状态、运行状况和广告等。

WSDM支持两种关系：资源间关系的简单信息，以及拥有自己特性和行为并可通过Web服务直接访问的信息。WSDM为公司或公司之间提供了商业系统综合管理的基础。这种发展将最终消除对特殊管理基础结构、管理员和客户商业系统管理应用综合的需求。

9. CBSD

CBSD(Component-Based Software Development，基于构件的软件开发)是一种基于分布对象技术，强调通过可复用构件设计与构造软件系统的软件复用途径。基于构件的软件系统中的构件可以是COTS(Commercial Off the Shelf，商用现货产品)构件，也可以是通过其他途径获得的构件(如自行开发)。CBSD体现了"购买而不是重新构造"的原则，将软件

开发的重点从程序编写转移到了基于已有构件的组装，以更快地构造系统，减轻用来支持和升级大型系统所需要的维护负担，从而降低软件开发的费用。

Web应用开发对灵活性、适应性、健壮性等方面的要求，提出了对基于构件的Web应用开发的要求。基于构件的Web应用开发方法的难点在于构建一个标准的数据结构、结构协议和程序体系结构。Web页面通过调用函数来使用操作系统中所注册的构件。

4.3.7 小结

可以用于Web应用建模的方法一般基于传统方法，如E-R图，或者是在面向对象基础上进行加强，如UML。目前的建模方法一般又是在早期Web建模方法上进行了增强。

建模方法归纳起来主要分为如下几种不同类型。

(1) 面向数据方法：基于E-R模型的方法，起源于数据库系统，主要关注数据库驱动的Web应用，如RMM、Hera、WebML以及OntoWebber和ADM。

(2) 面向超文本方法：关心Web应用的超文本特性，起源于超文本系统，如HDM、W2000、HDM-lite和WSDM。

(3) 面向对象方法：基于OMT或者UML，其中UML获得更大青睐，如OOHDM、UWE、OOWS、OO-H和WebComposition。

(4) 面向软件方法：将Web应用看做传统软件开发，使用沿着软件工程的技术，如基于UML技术的WAE及其改进版WAE2。

有些方法针对超文本而发展起来，对业务建模能力不足，而那些从面向对象建模发展起来的建模方法又对Web应用特殊性缺乏合适的描述。通常把Web应用模型看做是和传统软件系统一样，或者在实现层次上定义一些扩展，但都还不能完全满足Web应用开发方法应具有的特性。这也使得这些方法缺乏工程应用。要使得方法本身足够完善，除了要有强有力的工具做支持，还需要这些Web应用开发方法本身逐步完善并解决其中所存在的缺陷，而且还要从易学易用、支持能力自动化生成和转换方面做大量的改进，从而提供强有力的工具支持和工程化支持。

对Web应用开发而言，Web应用的模型有助于它们对系统的需求以及系统的架构和功能进行沟通，而一个好的建模工具对Web应用建模起着至关重要的作用。读者如果对相关内容感兴趣，可查阅相关资料，此处不再赘述。

4.4 功能需求建模

如第3章所述的各种Web应用需求识别、分析、描述、评估以及管理方法，用例是较好的功能需求建模方法，由用例图进行图形化描述。用例图可以直观地将功能需求从参与者的视角将Web应用的功能建模成一系列用例，这些用例从参与者(人和其他系统)的角度描述Web应用的需求。而且，用UML活动图可以对功能需求进行进一步细化。当用例的业务逻辑复杂时，例如一个用例可能被执行为一个Web服务或者一系列活动序列，再用活动图表达较为合适。

4.4.1 绘制用例图

用例经常使用用例描述的方法将其分解成一些步骤。用例描述采用文本的形式进行，外部角色是指与系统交互的人或外部系统，或是与系统有联系的其他设备。如果将一个角色称做一个类，则一个用户可以看成是该类的一个实例(对象)。一个用户在系统的不同场景中，还可以担任不同的角色，系统的一个用例可以看成是由一个或多个角色参与的场景。

Web应用需求的特色之一是其导航功能，导航允许用户通过超文本导航，从而发现结点。本章所采用的建模方法UWE将导航用例和功能用例构建在一个用例模型中，采用stereotype «navigation»来表示超文本用例与普通功能用例的区别。

绘制用例图时，首先需要建立模型，它定义了系统的主要功能和系统边界，完全是从系统的外部观看系统功能，并不描述系统内部对功能的具体体现，是其他视图的核心和基础。通过用例建模，描述对系统感兴趣的外部角色及其对系统的功能需求。在用例图中，角色代表触发系统功能的用户或其他系统，用例代表具体的功能描述。

1) 确定系统参与者

使用用例来分析系统，首先要确定系统的参与者。参与者是指所有与系统直接交互的人或事物，向系统输入或使用系统的某些功能，但是不属于系统。参与者可能是某个系统或一类人等。所有的Web应用至少会有一个参与者是人，至少有匿名用户。通过对系统的需求分析，可以确定系统的主要参与者。开发人员可以通过回答以下的问题来寻找系统的参与者。

(1) 谁将使用该系统的主要功能?

(2) 谁将需要该系统的支持以完成其工作?

(3) 谁将需要维护、管理该系统，以及保持该系统处于工作状态?

(4) 系统需要处理哪些硬件设备?

(5) 与该系统交互的是什么系统?

(6) 谁或什么系统对本系统产生的结果感兴趣?

在新闻系统中，我们识别出4类参与者：用户(User)、新闻撰稿人(Author)、新闻审稿人(Reviewer)以及系统管理员(Administrator)。在对参与者建模的过程中，开发人员必须要牢记以下几点内容。

(1) 参与者对于系统而言总是外部的，因此它们可以处于人的控制之外。

(2) 参与者可以直接或间接地与系统交互，或使用系统提供的服务以完成某件事务。

(3) 参与者表示人和事物与系统发生交互时所扮演的角色，而不是特定的人或者特定的事物。

(4) 每个参与者需要一个具有业务一样的名字，如新闻审稿人。

(5) 每个参与者要有简短的描述，从业务角度描述参与者是什么。

(6) 一个人或事物在与系统发生交互时，可以同时或不同时扮演多个角色。

(7) 和类一样，参与者可以具有表示参与者的属性和可以接受的事件，但使用得不频繁。

2) 获取用例

用例所描述的是系统的功能实现，它揭示了人们如何使用系统。

识别用例最好的方法就是从分析系统的参与者开始，考虑每一个参与者是如何使用系

统的。使用这种策略的过程中可能会发现新的参与者，这对完善整个系统的建模有很大的帮助。用例建模的过程是一个迭代和逐步精化的过程。新闻系统中，一般用户使用系统浏览新闻(Browse News)，搜索新闻(Search News)，登录(Login)、注册(Register)、评论(Comment)；新闻投稿者完成投稿(Submit News)，浏览评审结果(Browse Review Result)；审稿人评审新闻(Review News)，查看已提交的新闻(List Submitted News)，如果需要单独或者在查看已提交的新闻过程中浏览接受和拒绝的新闻稿(list accept reject news)；管理员维护系统(Maintain System)和发布新闻(Publish News)，其中发布新闻包含浏览接受和拒绝的新闻稿。在识别出的这些用例中，浏览新闻、查看评审结果、浏览已提交新闻稿、浏览接受和拒绝的新闻稿都是导航用例，使用 stereotype << navigation >>进行表示。新闻系统的全局简化用例图如图 4.7 所示。

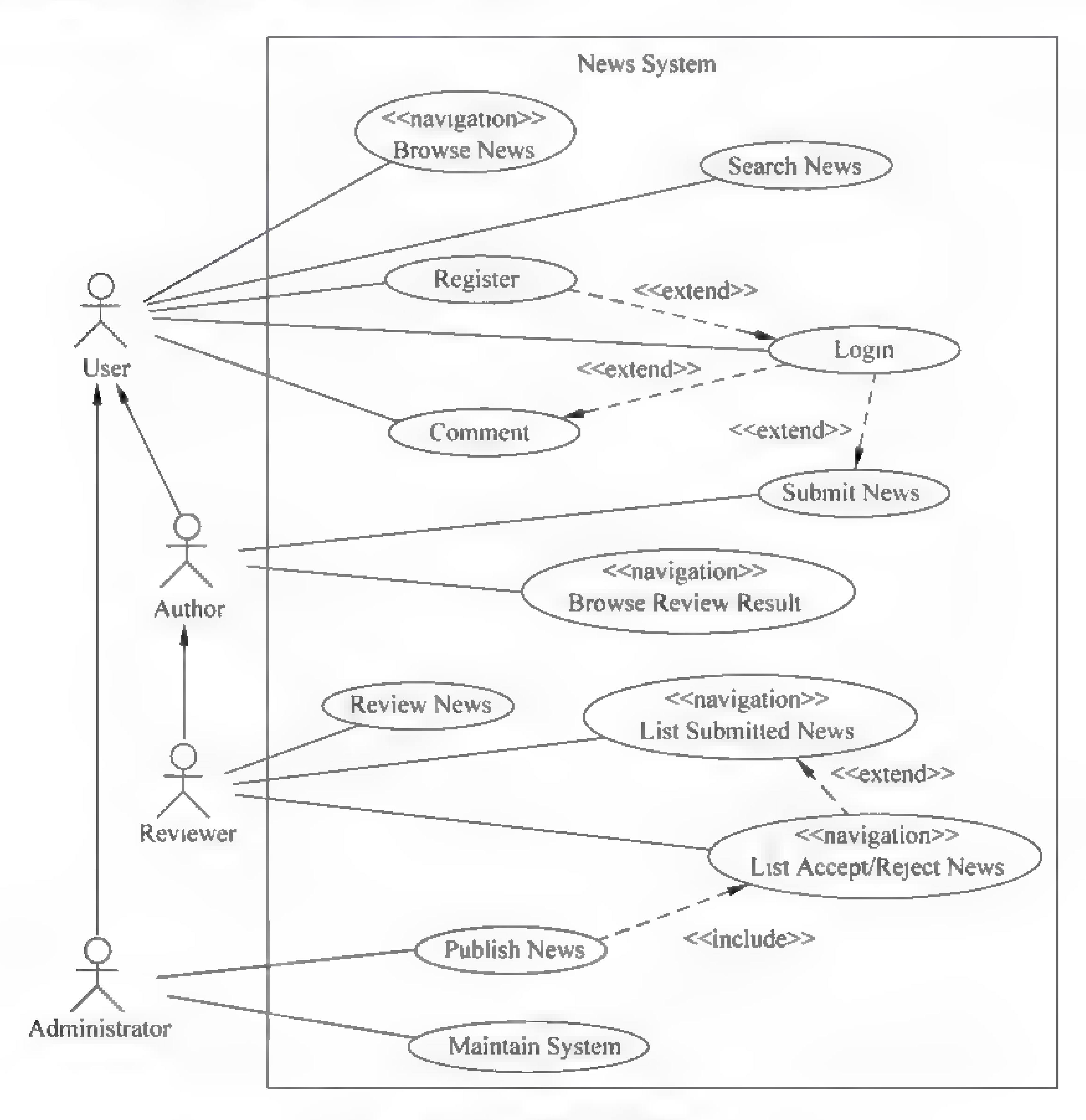


图 4.7 新闻系统用例图

在识别用例的过程中，通过回答以下几个问题，建模人员可以获得帮助。

(1) 特定参与者希望系统提供什么功能？

(2) 系统是否存储和检索信息？如果是，由哪个参与者触发？

(3) 当系统改变状态时，是否通知参与者？

(4) 是否存在影响系统的外部事件？

(5) 哪个参与者通知系统这些事件？

4.4.2 绘制活动图

绘制好用例图之后，对其中表达复杂业务逻辑的用例进一步进行建模，采用UML活动图进一步精化。新闻系统的新闻投稿、新闻审稿、新闻审核以及系统管理等用例，都是相对比较复杂的用例。以撰稿人进行新闻投稿为例说明其精化后的活动图，如图4.8所示。

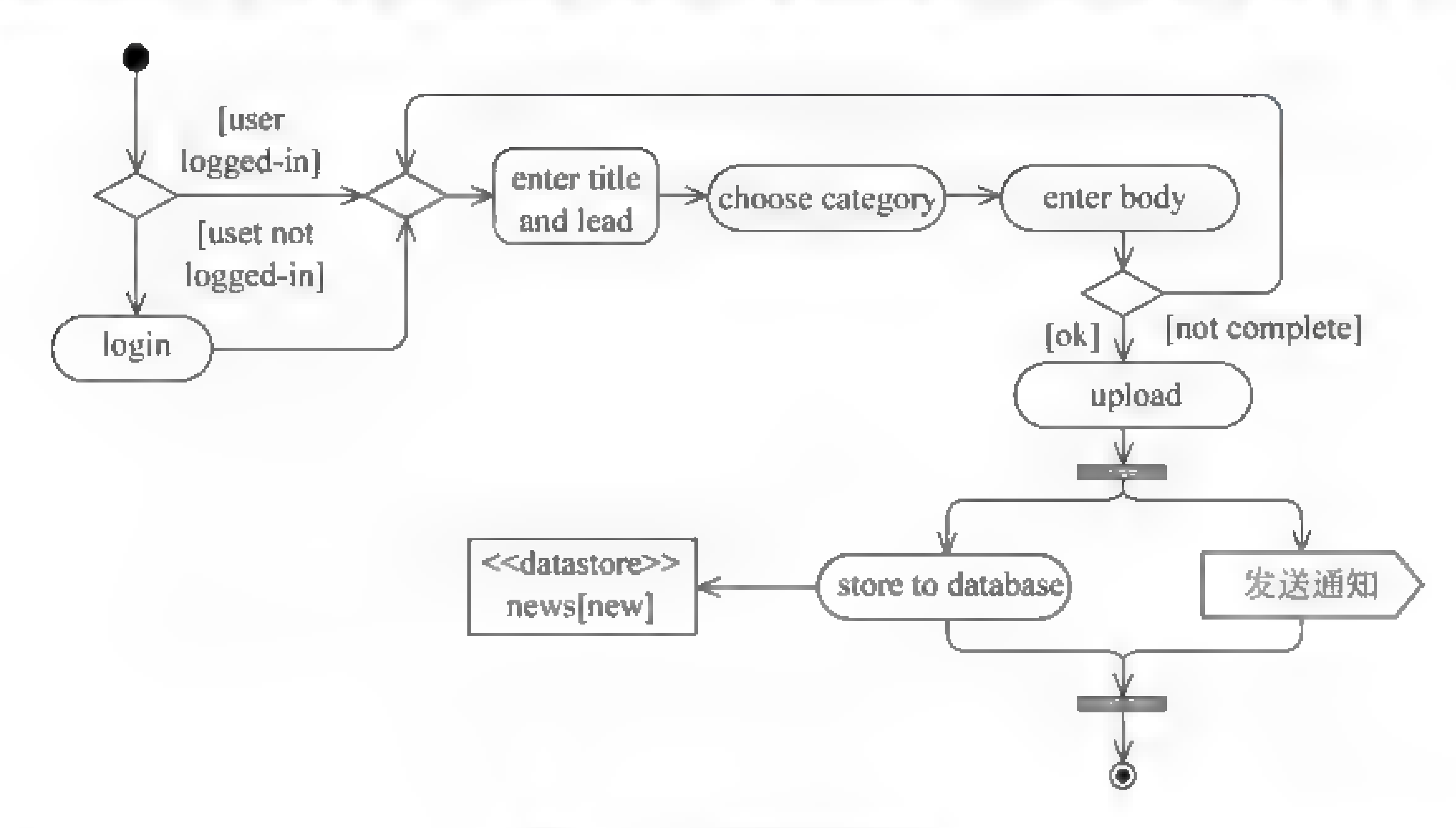


图4.8 新闻投稿活动图

图4.8中展示了如下所述的业务内容。如果用户是新闻撰稿人，则登录到新闻系统，提交新闻稿。提交新闻稿时需要输入新闻标题和导语、选择新闻所需类型、新闻摘要以及新闻内容(可以是在线编辑，也可以用附件上传)，上传上述内容后存入数据库同时发送提交成功通知。

4.5 内容建模

内容建模的目标是将从需求工程中决定的信息和功能需求，转换为模型。Web应用所提供的信息是一个成功的Web应用的最重要的因素之一，它决定了用户能看到的内容。这里需要考虑Web应用的超文本特性及其表示，而且要确保存在的信息是没有冗余并且可以重用的。

对静态Web应用来说，对内容进行数据建模就足够了。但是，对于复杂Web应用，就需要添加对行为方面的建模了。内容建模包括创建问题域模型，问题域模型包括静态和动态两个方面。在内容建模时考虑如下两个方面的内容。

(1) 以文档为中心的特性和多媒体特性。在内容建模时需要考虑所有不同的媒体格式和信息的结构。

(2) 与现有数据和软件的集成。许多Web应用是以已经存在的数据库和软件组件为基础所构建的，这些数据库和组件起初并不是为Web应用所创建。内容建模必须满足潜在而矛盾的目标，即内容建模应该满足Web应用内容需求具有很好的扩展性，并且包含现有

的数据结构和软件组件。

4.5.1 静态建模

内容建模的静态模型用类图表示。新闻管理主要涉及新闻、新闻分类信息、评审信息、评论和各类用户等内容，分别抽象为：News、Category、Review、Comment 以及 User。图 4.9 所示为其简化的类图。

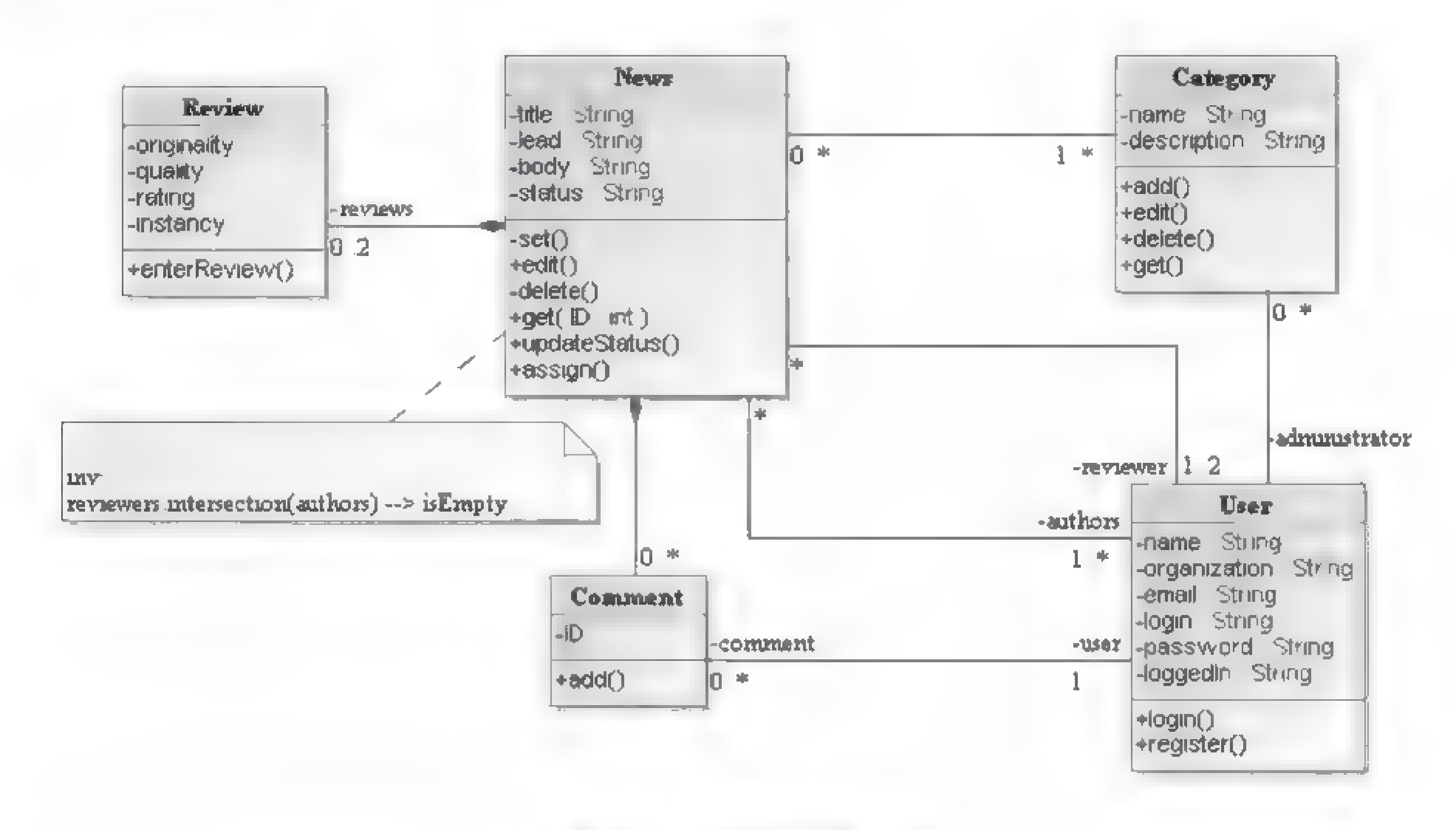


图 4.9 新闻系统类图

图 4.9 中所示为将新闻系统建模为具有多种新闻类型，如时政、体育、影视等；一条新闻可以隶属多个分类；用户有普通用户、新闻撰稿人、新闻审稿(政审和内容审核)人以及新闻管理员。一篇新闻稿需要经过内容和政审两个审核者的审核，并且如果他们自己也是撰稿人，则不可以审核自己提交的新闻稿。用户可以对新闻进行评论，一条新闻可以有多条评论，一个用户可以对一条新闻有多条评论。

图 4.9 所示类图将作为后续超文本建模和展示建模的基础。

4.5.2 动态建模

内容建模的动态建模是将静态内容模型中有关类的状态转换进行建模。UML 状态图用于建模内容的动态性。以新闻系统中新闻稿为例，一条新闻稿在提交之后，需要由新闻审核人进行内容和政治审核，如果其中有审核没通过，给撰稿者发送邮件通知稿件被拒；都通过后，稿件被录用，由新闻管理员进行格式审查，如果审查有问题，返还给撰稿者进行修订；审查通过后，安排在线发表并发送邮件通知撰稿者稿件已发表。可以看出一条新闻稿具有多种不同的状态，采用状态图进行建模需要经过的状态以及状态转换条件或时间如图 4.10 所示。

图 4.10 展示了新闻稿提交后，从审核到最后发布之间的状态迁移。提交之后首先为已提交(submitted)状态；给内容审核人和政审人进行审核时为正在分配(under assignment)，其

中按每个时钟点 at(clock) 主要是考虑系统自动按角色进行分配；两个审核人都接受审核任务之后，进入正在审核状态(under review)；计算每个审核人的评审结果后未达到录用标准(evaluation<threshold)，稿件就进入被拒(rejected)状态；达到录用标准(evaluation>=threshold)，稿件进入被录用(accepted)状态；由新闻管理员进行格式审查，即格式审查状态(format review)；审查通过后，进入在线发布状态(inPublish)，发布完成，状态转换结束。

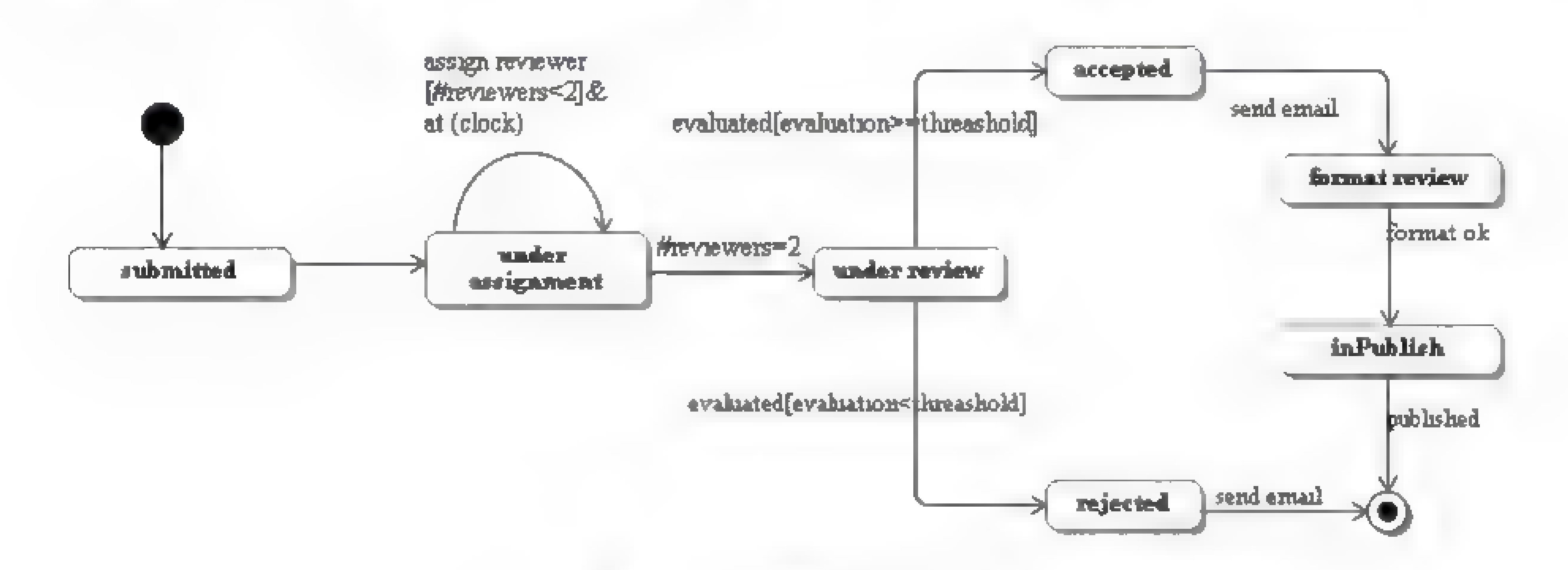


图 4.10 新闻类的状态图

4.6 超文本建模

超文本的非线性特性是一个需要特别注意的问题。因此，在对 Web 应用进行超文本结构建模的时候需要非常小心。通过使用合适的访问结构可以避免无效链接等问题，从而高效地建模超文本。

超文本建模的目标是通过 Web 应用的内容构建导航，因此也是导航建模。它主要包括动态和静态两个方面的结果。静态方面是超文本结构模型，也称为导航结构模型，用于表述超文本的结构。这一类内容模型可以通过导航来访问。动态方面是超文本访问模型，使用访问模型中的访问元素精化超文本结构模型。某些建模方法会使超文本模型对内容模型存在或多或少的依赖。一是类层的依赖，如内容模型的哪个类在超文本模型中的哪个结点；二是实例层的依赖，如内容模型中的哪个对象集构成超文本模型中的哪个结点。

4.6.1 静态建模

超文本建模的静态建模主要是强调超文本的结构。和内容建模采用类图进行建模不同，超文本建模需要引入特定的模型符号。超文本建模主要基于超文本的概念，即结点(页面或文档)和结点之间的链接，用来表示在 Web 应用中哪些对象能够被访问以及对象之间的导航关系，一般用导航类图来表示。

超文本建模一般以内容模型为基础，将其中的类和对象在超文本中表示为结点。超文本模型通常被当作内容模型的一种特定视图，因此称为导航视图。结点在内容模型中可选择一个或多个内容对象。某些方法还定义了根据内容模型中的关系导出链接的转换规则，另外根据设计需要，添加一些链接。OOHDM 中超文本模型和内容模型之间相互独立，根

据定义的需求情景中直接识别导航需求而建立链接。

不管采用哪种方法，我们都可以创建出内容模型之上的超文本视图，即超文本结构模型。例如，如果我们考虑超文本结构模型中的访问权限，就可以构建出个性化的超文本视图(个性化建模在4.8节中详细说明)。

在新闻系统中，要求为新闻撰稿人、新闻审核人和系统管理员等用户角色构建不同的超文本视图。考虑建模系统管理员视图的超文本结构模型，管理员可以查看所提交的新闻稿，可以访问接受发布的和退回的新闻稿，还可在特定情况下查看系统内用户的资料。图4.11展示了采用UWE建模的结果，其中采用UML stereotype << navigation class >>来表示在超文本模型中的结点类，目的也是为了和内容类进行区分。带箭头的关联线表示链接，采用stereotype << navigation link >>进行表示，而如果是一个业务处理过程的起始结点则用stereotype << process link >>来表示，如果是指向不属于当前应用的结点，则用stereotype << external link >>来表示。

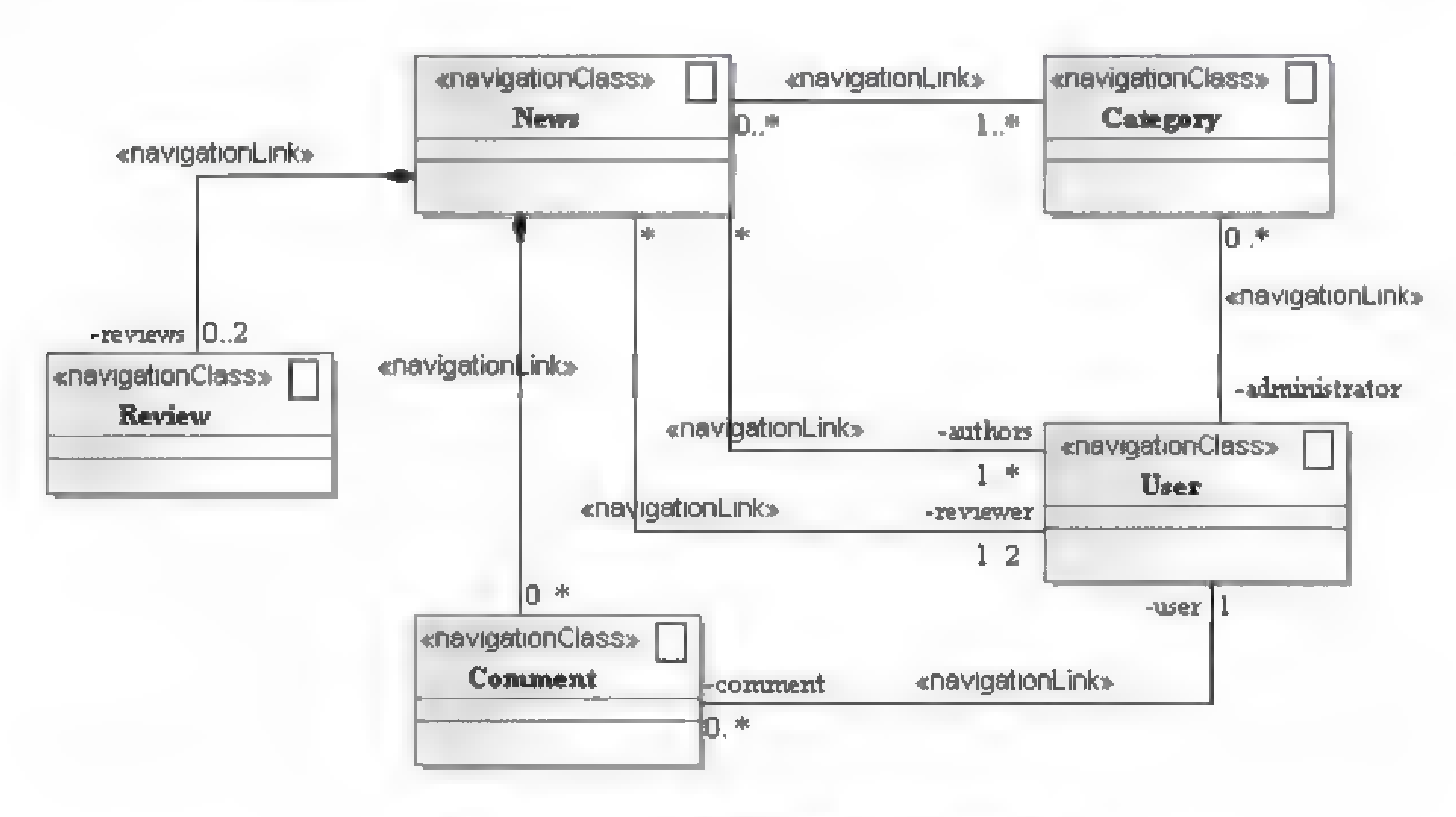


图4.11　管理员视图的超文本结构模型

其他一些方法中也提供了有关链接的建模支持。例如HDM中的structural link、perspective link和application link表达不同的语义；WebML中的contextual link和non-contextual link表达导航时是否信息传递，页内链接和页面间表达展示层页面中，超文本结点的分布。在OO-H中，定义了I-link、T-link、R-link、X-link和S-link。有关建模方法的详细信息参看4.3节。

4.6.2　动态建模

超文本建模的静态建模目前还不能够描述如何通过导航访问到结点。要使用户能够导航到他们需要的结点，需要一些导航和方向性的帮助，这些导航和方向性帮助通过访问结构来对超文本结构模型进行精化，以体现超文本模型的动态方面的建模。访问模型主要强调超文本和访问元素，或称为导航元素，导航行为通常并不显式表示出来。访问结构的设计模

式，也称为“超媒体设计模式”或“导航模式”，有助于很大程度上提高超文本模型的质量。这些模式描述了页面如何把特定行为产生的结果通知用户，用户如何基于使用环境和自己的期望扩充内容，如何最好地描述链接所暗示的目的地，如何把正在进行的交互行为以及界面相关的问题告知用户。有关导航设计模式的更多信息，参见 German 等人的文献和网上更多的资料。

对于 Web 应用而言，导航的作用无可替代。首先，导航可以提供 Web 应用概念地图，通过导航，用户可以了解到整个 Web 应用的轮廓；其次，导航可以给出有关用户当前位置的反馈；再次，导航可以帮助用户找到他们想要的东西，有助于用户毫不费力地在 Web 应用内检索内容；最后，通过导航，用户能意识到 Web 应用上对于他而言还可能有用的东西。

在新闻系统中，如果想通过审核人导航到分配给该审核人的一篇新闻稿，就必须识别这篇新闻稿。这一功能可以通过列出所有新闻稿的方式实现，这样的选择列表也称为索引，索引访问结构允许用户从同类对象列表中来选择一个对象。相反，菜单则允许用户访问不同类型的结点，或者更多的菜单（子菜单）。其他的访问结构有向导和查询。向导（Guided Tour）允许用户顺序访问一些结点，查询（Query）则允许用户搜索结点。大多数建模方法为最常使用的导航模式提供专用的模型元素，如 home（主页）指向 Web 应用的主页，landmark（地标）指向一个可以从所有结点访问到的结点。一些访问结构可以被自动加入超文本结构模型，例如，索引可以在允许访问一个结点对象集时被自动加入。

图 4.12 是图 4.11 所示结构模型的一个简化访问模型示意图，是新闻系统中超文本结构模型的新闻管理员视图。注意：一个链接的默认多重度是 1。管理员必须访问所有新闻稿、审核人、撰稿人和用户。为了访问一篇特点的新闻稿，需要唯一 ID；也可以通过新闻标题搜索新闻稿。UWE 使用 UML stereotype « menu »、« index »、« query »和« guided tour »等来明确表示菜单（如 MainMenu）、索引（如 ReviewingStatus）、查询（如 Search）和向导等访问结构。« process link »和« navigation link »分别表示处理链接和导航链接。

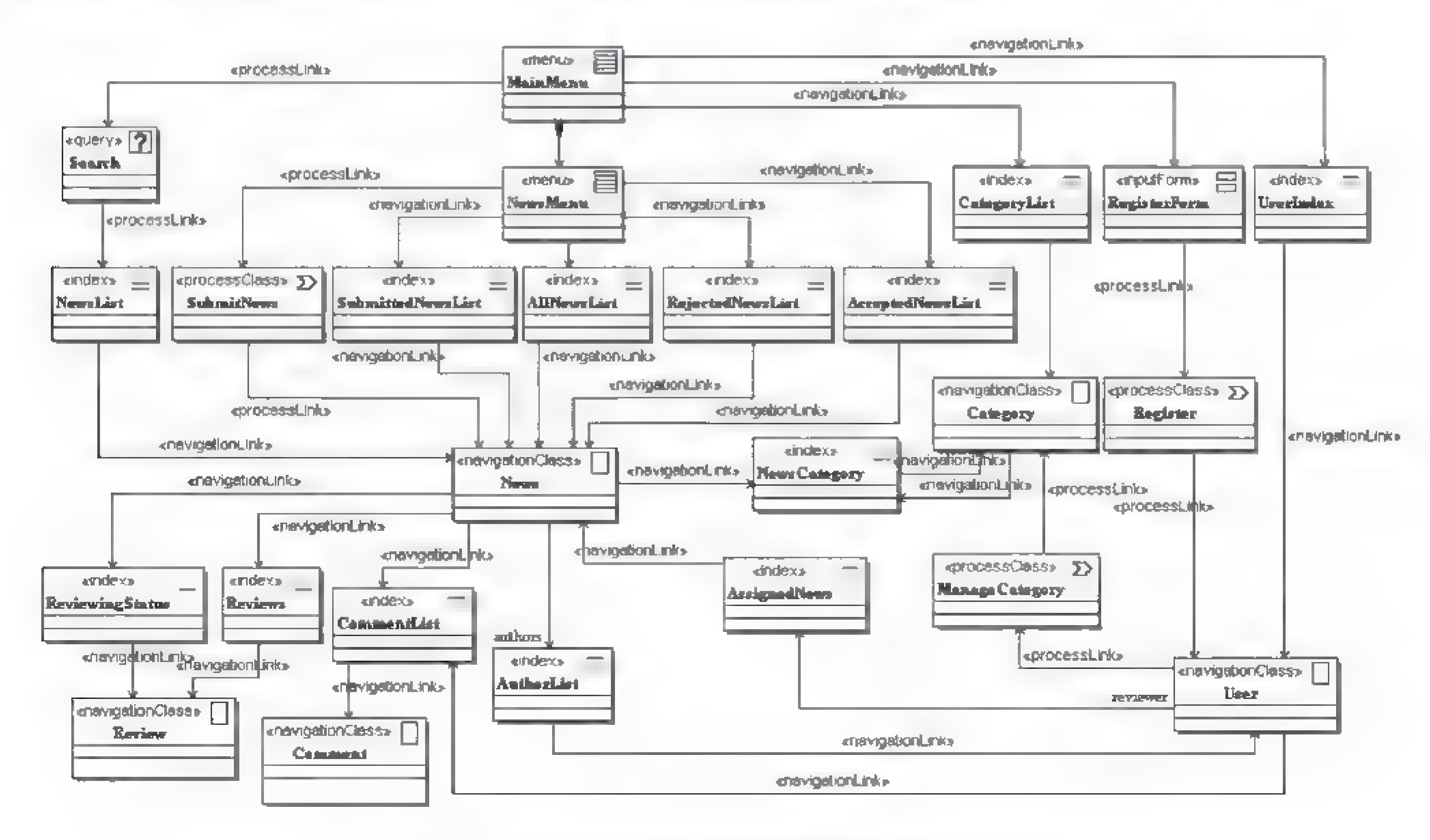


图 4.12 图 4.11 所示结构模型的简化访问模型

对 Web 导航能力的建模还有其他对 UML 的扩展方法，如对用户登录操作的模型如图 4.13 所示。

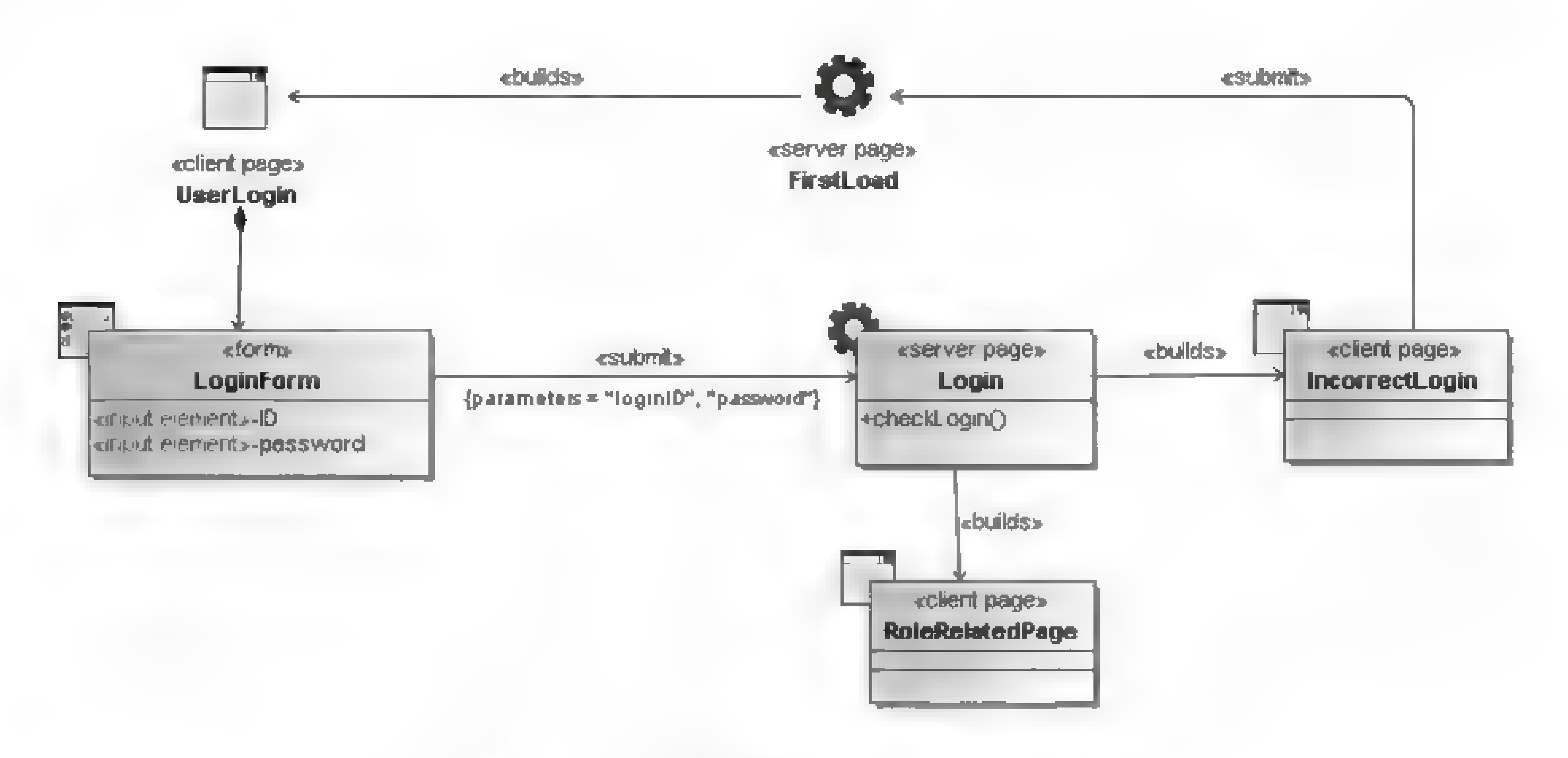


图 4.13 用户登录新闻系统的 Web 模型

4.7 展示建模

和传统软件工程一样，Web 应用的展示建模主要针对 Web 应用的用户界面。其不同点在于 Web 应用的主要表现形式是 Web 页面。Web 页面是 Web 应用的基础，用户所看到的信息以 Web 页面通过浏览器展现出来。因此 Web 应用的展示建模主要是对 Web 页面进行建模。

对 Web 应用进行展示建模的主要目的是 Web 页面和页面内元素的结构以及用户界面的交互行为应该简单且可自我解释，同时考虑 Web 应用的本身的交互和展示职责。展示建模包括两个方面：一是通过建模页面中的通用元素（如页眉和页脚）产生一致的展示概念，并且展示每个页面的组成以及其中包含的字段、文本、图片、表单等内容；二是描述用户界面的交互特性，例如单击某个按钮可以激活应用逻辑的某个功能，同时要考虑提供给用户必要的导航路径、用户访问历史等信息以防止用户迷失在复杂的应用之中。Web 应用的另一个重要特性是图形化布局设计，通常会使用一些可视化图形设计工具，或者实现一些原型页面。

类似于超文本模型和内容模型的映射关系，超文本元素和展示元素之间也具有映射关系，这通常是因为结点的每个实例将在展示层上显示。用户触发的交互并不局限于展示层本身，因此，在建模的时候需要考虑这些交互和其他链接的依赖关系，这些依赖可以是内容层的对象形式和应用逻辑，也可能是超文本层的导航。

4.7.1 静态建模

在 Web 应用中，展示层界面的组成及组织本质上是应用系统的 Web 页面的静态结构编排。针对 Web 页面及其包含表单、文本、图片、按钮等元素的特点，将建模元素分为三个

层次：表示页面(<< page >>)、表示单元(<< presentationGroup >>)和表示元素(<< text >>、<< textInput >>、<< inputForm >>、<< anchor >>和<< button >>等 stereotype)。表示页面描述用户看到的最大单位，可以包含多个表示单元；表示单元用于将一些页面元素组织在一起，表达页面的逻辑片段，表示超文本模型中的一个结点；表示元素则是表示模型的基本单位，表示文本、图片、音频、按钮等结点中的信息。

在 UWE 中，通过基于嵌套 UML 类图的形式来表示页面中的组成关系。图 4.14 所示为新闻系统中新闻页面(NewsPage)的简单展示模型示意图。

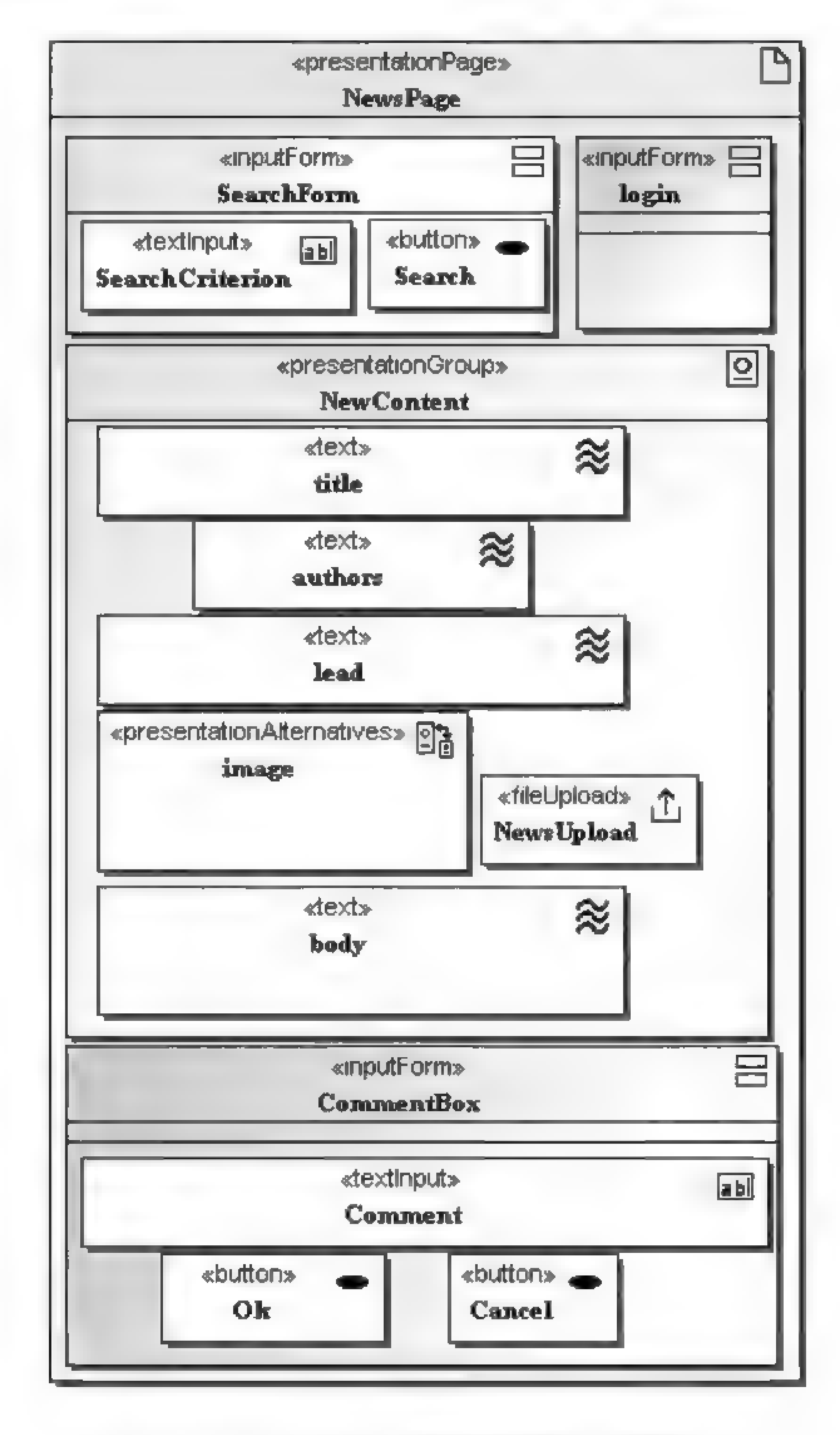


图 4.14 新闻系统中新闻页面的简单展示模型

图中展示了一些页面元素，如搜索表单、用户登录表单。一般用户可以通过单击发表评论按钮发表其对当前新闻的评论。新闻的内容作为展示组(<< presentationGroup >>)，其中

包含标题、作者、导语和新闻正文等文本(<< text >>)元素；还可以表达可候选的内容(<< presentationAlternatives >>)，如图 4.14 中的 image，在提交稿件时采用上传文件(<< fileUpload >>)的方式。

4.7.2 动态建模

展示层的动态方面主要是 Web 页面的行为，即用户为了完成某项任务通过页面和 Web 应用之间所进行的交互行为。交互行为可以通过行为图来进行建模。比如在新闻系统中，审核人(内容或政审)交互式地获取到分配给自己的新闻稿(如图 4.15、图 4.16 和图 4.17 所示)。一般情况下，用户和 Web 应用的交互不仅涉及展示层，根据交互类型的不同，还可能也需要与超文本层和内容层进行交互(参见图 4.16 和图 4.17)。一个审核人通过新闻主页中的导航条选择所分配给自己的新闻稿索引列表，这一动作需要在内容层根据身份进行组成，审核人再导航选择一篇新闻稿进行审核，然后再显示所选中的新闻稿的详细新闻审核视图。

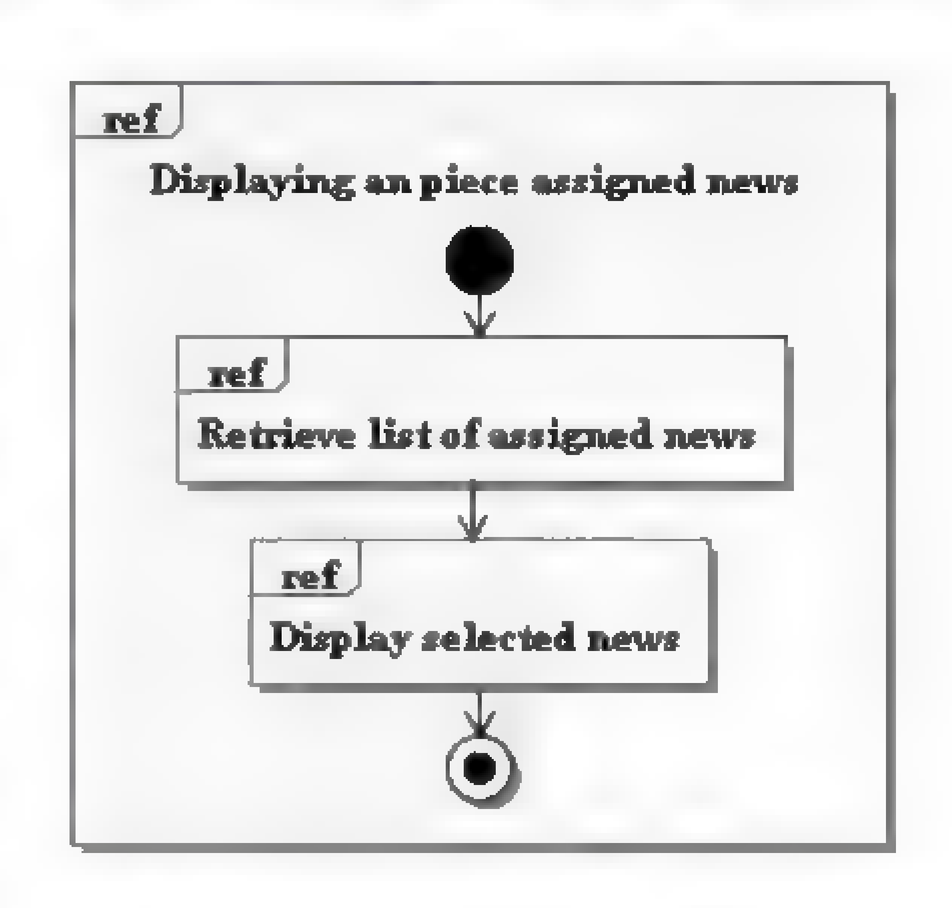


图 4.15 显示所分配新闻稿交互图

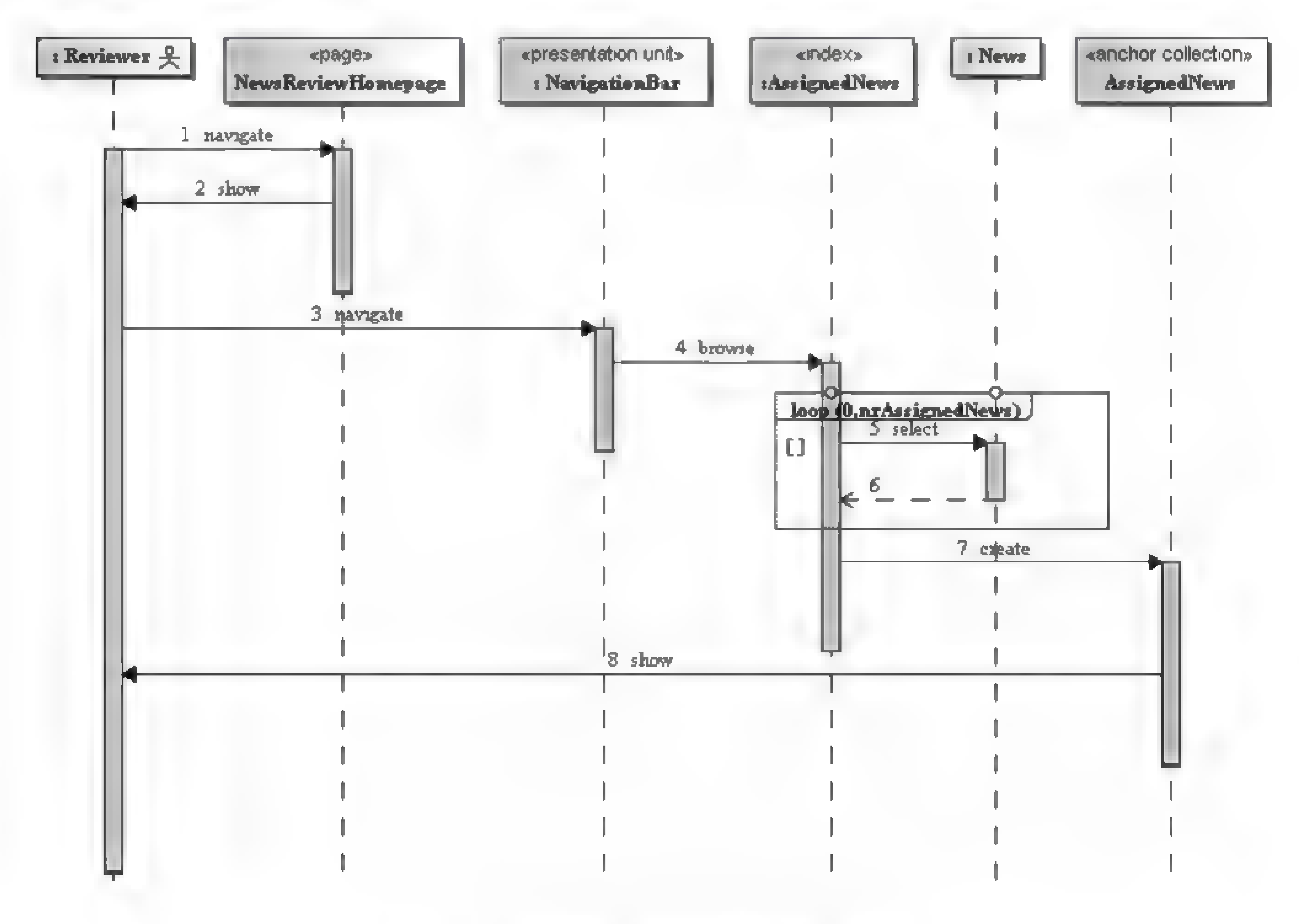


图 4.16 获取所分配新闻稿列表序列图

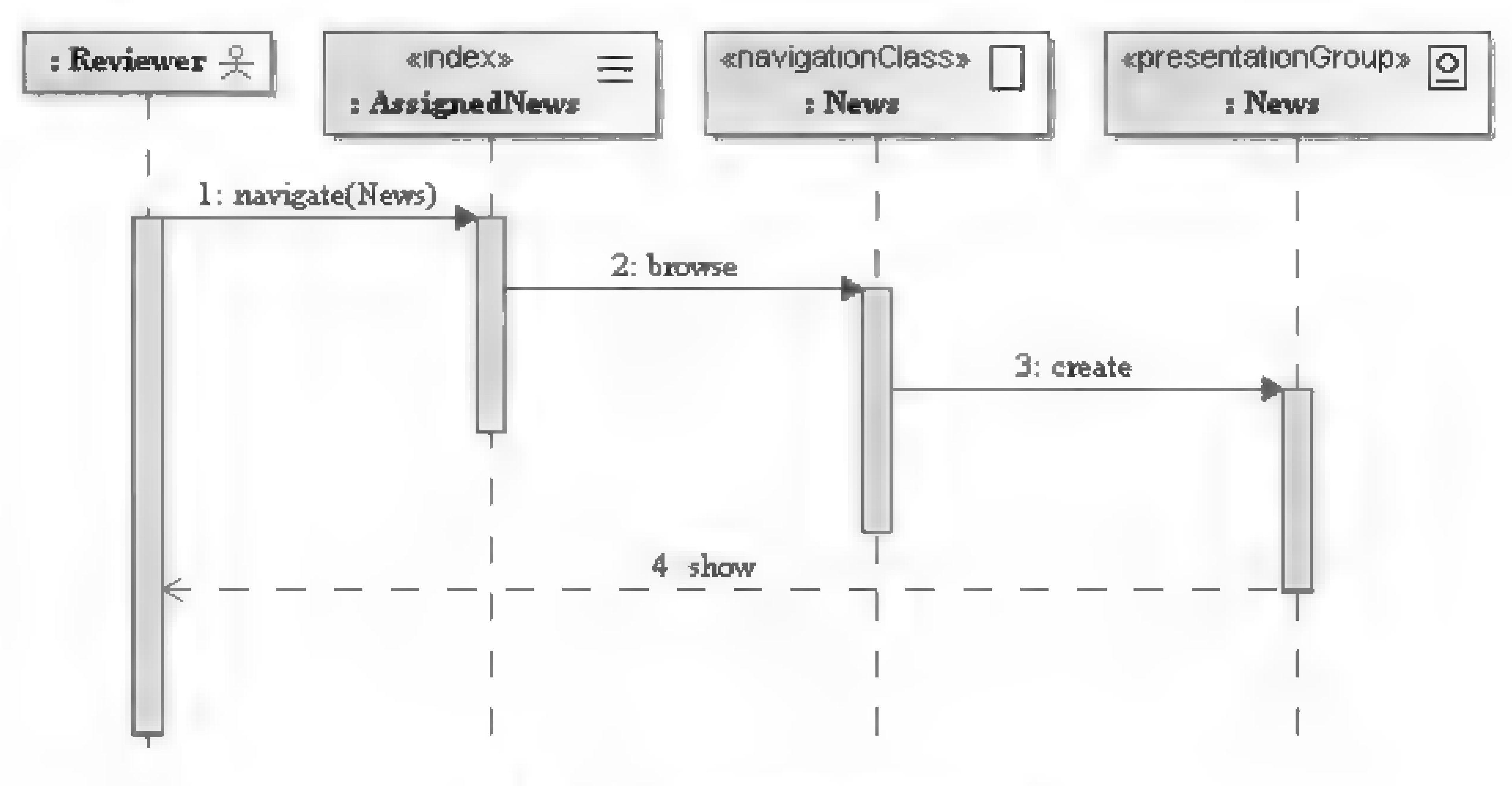


图 4.17 显示所选新闻稿序列图

4.8 适应性建模

Web 应用的普适性越来越重要，使用越来越广，因此，适应性是 Web 应用越来越重要的特性。可适应性 Web 应用根据用户上下文(Context)特性，给用户提供合适的页面。适应性建模需要考察 Web 应用的使用场景，如什么时间该进行什么适应性调整。比如，越来越多的 Web 应用提供手机访问支持，而手机有类似于 iPhone 的功能强大、体验较佳的高端产品，也有比较一般的低端产品，上网带宽也可能不同。不同用户(或组)对信息的需求不同，对 Web 页面外观的喜好也不同。用户的满意度、使用体验对许多 Web 应用非常重要，如电子商务系统、网络游戏等应用，用户量很大程度上决定着企业的收益。而 Web 应用的特点是用户有很大的不确定性，任何人都可能成为 Web 应用的用户。考虑到每个用户(组)的不同特征与不同需求，Web 应用提供不同的 Web 页面和不同的功能，也就是适应性。

因此，需要尽早在建模阶段考虑上下文信息和 Web 应用的可适应性。适应性建模旨在显式地表达上下文信息，以及由上下文而产生的适应性。为了使 Web 应用能够适应个性化特性，就必须建模和管理用户的爱好和特性信息，将这些信息保存于用户配置(Profile)中。例如，为了使 Web 应用能够适应移动计算，就需要考虑设备情况、位置信息和传输带宽。然后，将用户配置信息保存于类图形式的上下文模型中。在运行时，上下文可以变化，如用户改变他们的喜好，或者在不同的地点访问 Web 应用。而这样的情况也就是我们为什么必须适应性调整 Web 应用。

目前的建模方法中，个性化通常并不能在模型上显式地表示出来，而且，在很多情况下，如 4.1.4 节所述，适应性影响 Web 应用建模过程的所有层。变化可以局限在一层内部，也可能影响几层。个性化建模交织在内容、超文本和展示多个模型中。对内容、超文本和展示模型分开考虑模型适应性，有助于提高可变性、灵活性和封装性。不过，目前大多数建模方法并不支持这种分开考虑，而且有时候这样的分开考虑本身就很困难。

这里,我们需要区分适应性与维护和再工程两个概念的不同。适应性建模考虑的是在建模时就能预知的上下文信息,可以认为上下文信息在Web应用运行时具有不同的值。相反,因为组织或者技术环境的变化而需要做出适应则属于维护或再工程活动。

考虑上下文信息的抽象级别,可以将上下文区分为物理上下文和逻辑上下文。物理上下文由用户各自的使用场景产生,如用户的登录名、登录地点等。逻辑上下文提供更多其他上下文信息,如用户的工作地点、家庭地点、工作时间和空闲时间等。这些上下文信息也可以通过外部资源提供,如GIS。

目前对个性化的建模仍然处于起步阶段,支持个性化建模的方法很少。早期对个性化支持的建模工具,如ContextTookit或NEXUS已经提出能够支持不同类型的物理和逻辑上下文信息的通用组件。WebML提供组和用户两个实体,并支持个性化建模。组实体代表一组有着共同特征的用户,用户表示的是单独的个体。每个用户和组都表示一个轮廓(Profile),是一个特殊的实体。Web应用设定一些规则,系统根据这些规则对用户进行分类,将组和站点视图关联起来。当一个用户访问系统时,系统根据分组规则判断用户归属的组,然后查找该组对应的站点视图,展现给用户。

对上下文进行建模可以分为静态适应和动态适应两种不同的方法。不管是动态还是静态,都要依赖于用例。

而UWE方法对个性化的处理则是通过面向方面(Aspect-oriented)方法。

4.8.1 静态建模

个性化建模的静态建模指静态适应,即根据上下文信息的各种不同变化集,创建面向结果(Result-oriented)的不同模型或模型变换。图4.11所示的超文本模型描述了新闻管理员用户角色的超文本结构,这种根据用户角色上下文进行静态适应需要考虑多种模型的变化以及可能的模型数量的增加。

4.8.2 动态建模

个性化建模的动态建模指动态适应。和静态适应相比,采用动态适应时要提交一些和上下文相关的内容、超文本和展示模型的转换规则。这些转换规则描述在运行时需要进行的变化。例如,采用ECA(事件/条件/动作)规则描述的动态转换规则,能够指定添加或删除模型元素,或者过滤实例,可以创建读者关注的新闻主题的新闻列表。

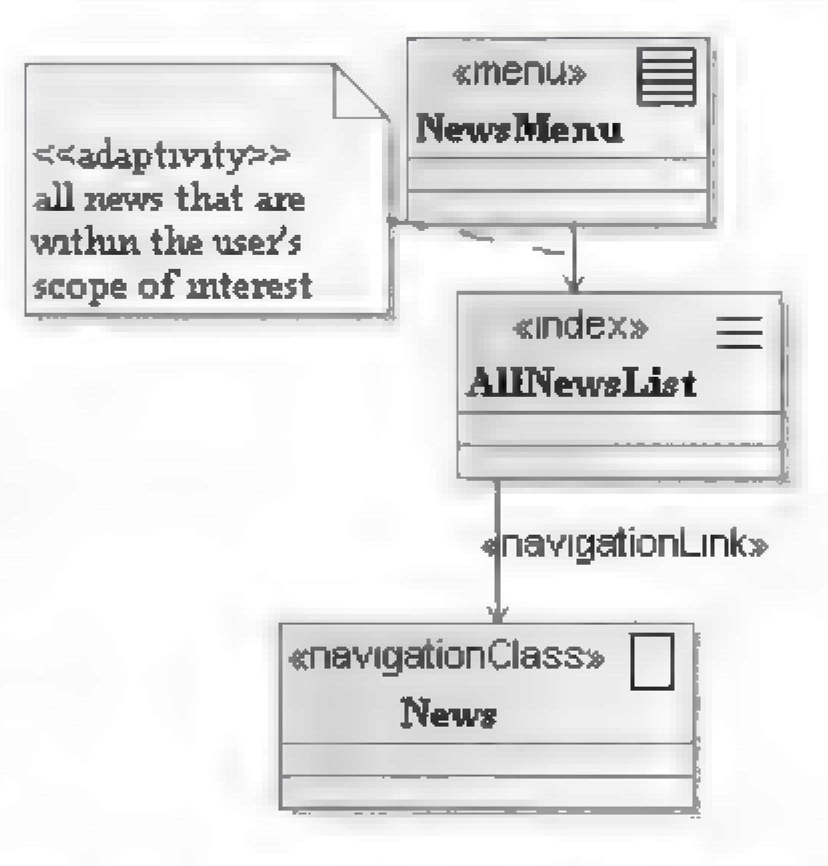


图4.18 超文本模型中索引的动态适应

图4.18和图4.19描述了新闻系统的超文本模型和展示模型是如何适应的。采用stereotype <<adaptivity>>注释,对要进行适应的类添加适应规则。图中非正式的说明部分可以采用如OCL(Object Constraint Language)的更加正式的形式化语言进行进一步精化。图4.18中显示了超文本结构中实现所有新闻都是用户感兴趣的个性化建模方式。访问结构的元素"InterestingNews"可以根据用户个性化新

闻主题的设置进行动态调整。图 4.19 中展示了展示模型中的元素如何根据转换规则进行动态适应，尤其是“NewsUpload”按钮应该只有 author 才可见。

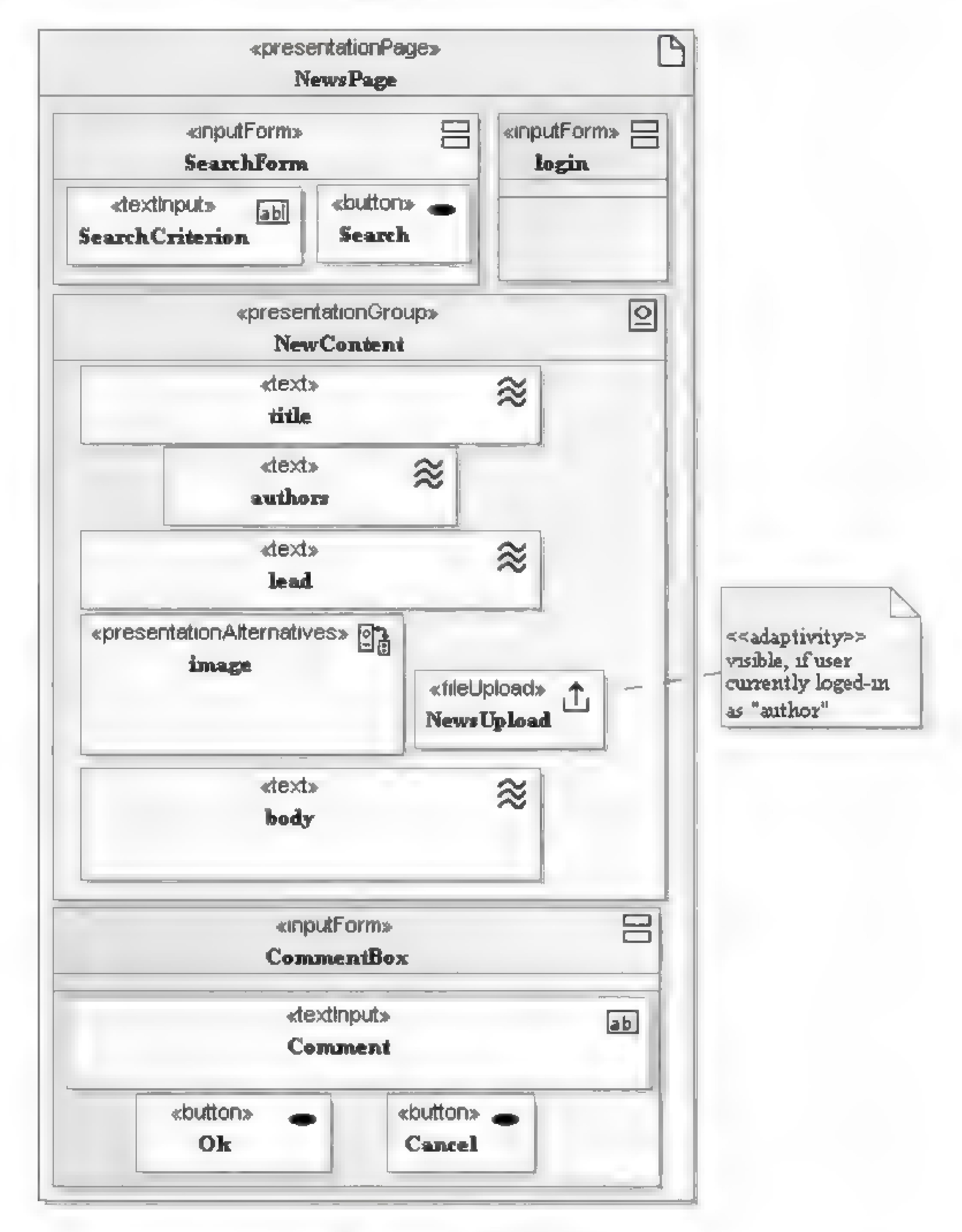


图 4.19 页面的动态适应

大多数现有方法中仍然采用为 Web 应用中每个需要进行动态适应的部分定义规则，如上例所述。

如 4.3.1 节中所述，UWE 采用面向方面建模(Aspect-Oriented Modeling，AOM)技术进行适应性建模，可以使得系统功能和个性化方面系统地进行分离，也可以减少冗余。UWE 区分内容、超文本和展示层的不同个性化建模。例如，链接可以被注释、排序、隐藏，或者根据用户当前状态或上下文模型动态生成。其实现方式是通过扩展 UWE 元模型，添加建模元素 <<Navigation Annotation>>。这个模型元素可以附加在任何链接上，图 4.20 给出了建模者如何使用该建模元素来添加一个在 advice 部分包含的属性(PresStyle)，以及横切点(crosscut)部分中包含的所有链接的方法。图 4.21 显示了 weaving 过程的结果。

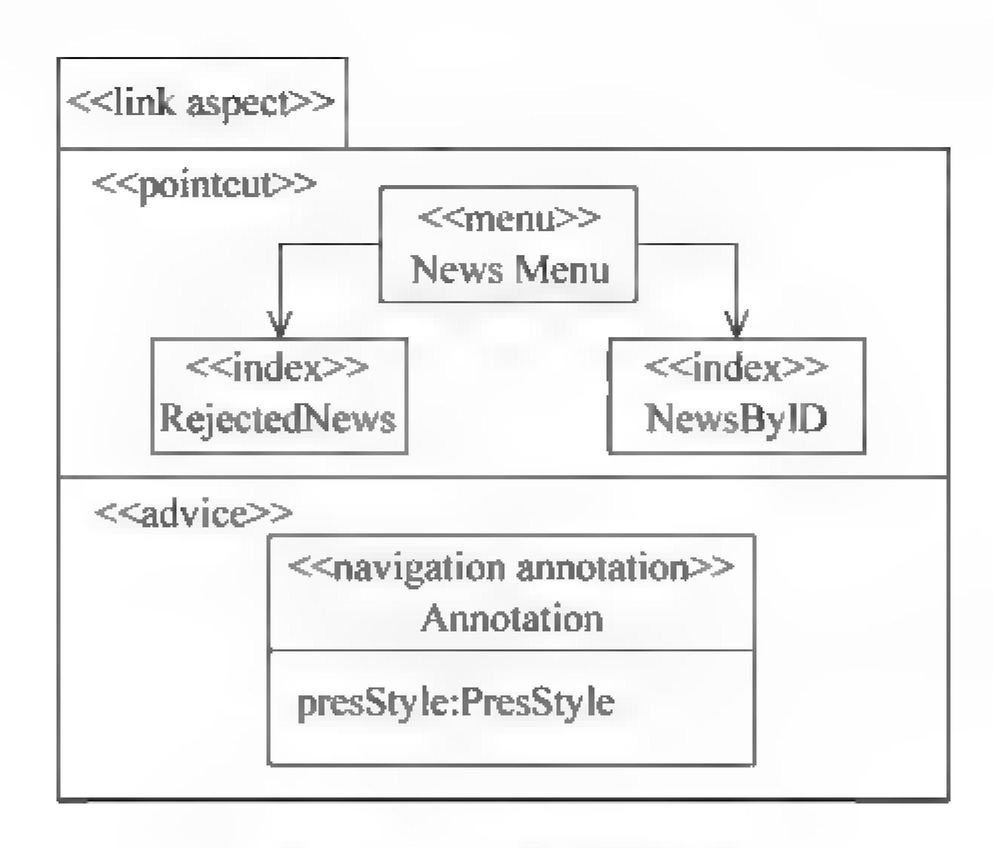


图 4.20 方面动态适应

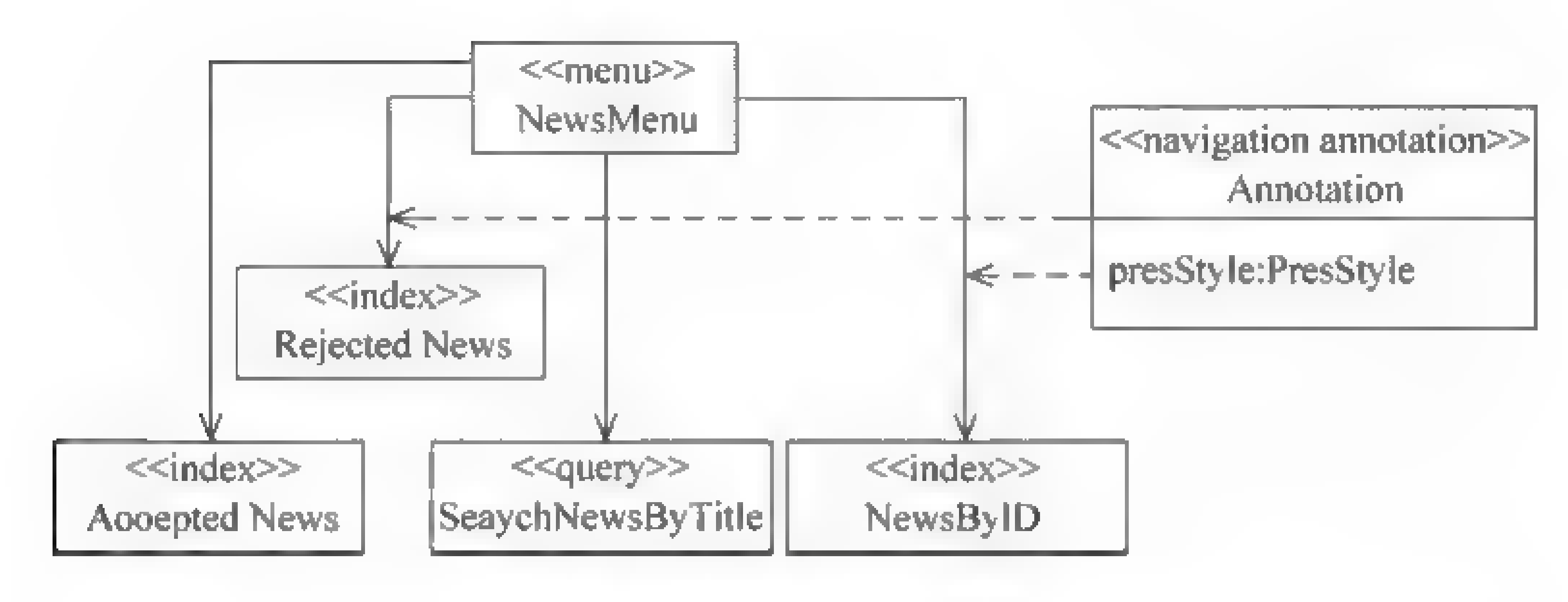


图 4.21 weaving 结果

动态适应可以通过包含由 OCL 所写的方面的 advice 部分的规则来完成。

4.9 总结与展望

Web 应用建模的特性包括层、方面、阶段和适应性 4 个方面。模型驱动开发的优势不断展现，也使得在短短的 10 年间，出现了许多不同的针对 Web 应用特性的建模方法。本章从模型驱动开发的角度出发，分别从需求建模、内容建模、超文本建模、展示建模和适应性建模的静态和动态两个方面对 Web 应用建模加以讨论，简单介绍了几种 Web 应用的建模方法和支持工具。

有些方法可能在后期发展中获得优胜，目前还很难说需要多久出现和 UML 建模语言一样的"统一 Web 建模语言(UWML)"。不过可以认为，类似 UML 作为符号语言的趋势在逐渐增强。哪种建模方法获得全胜，支持工具起着决定性作用。而且，MDD 有 OMG 的 MDA 和 QVT 标准支持，会对开发方法和工具起到很大影响。因此，工具不仅需要支持建模符号，也要支持模型驱动开发的开发过程。这也就意味着，建模方法需要定义清楚的指南和方法，以及需要在很大程度上考虑敏捷方法，而且，需要与自动生成相协调。

建模方法的发展，必须考虑 Web 应用运行平台和发布框架的不同、重用的建模、工作流建模等以满足 Web 应用中日益增长的事务处理需求，在尽可能早的建模阶段考虑 Web 应用的上下文，以满足不断增长的移动用户的需求。但是，也需要考虑在模型的复杂性增长的同时，质量可能会变成重要问题。基于模型的 Web 服务也一样面临新的挑战。

第5章 Web应用架构

架构展示系统的框架，描述系统的结构，从静态和动态两个方面描述系统如何划分为组件及其接口和相互关系，有助于人们对系统的理解。架构是系统的抽象视图，根据角度不同，可以将架构分为4种视图：概念视图、运行视图、过程视图和实现视图。架构也形成了系统从分析到实现的转换。

Web应用的质量很大程度上受制于底层架构，而架构受制于系统的功能需求、质量需求、开发团队经验和开发技术。架构不完整或者缺少了某些方面，使得Web应用的一些质量需求很难满足，甚至无法满足。Web应用的架构不合适经常会导致性能可维护性和可扩展性差以及可用性低等问题。Web应用的架构除了采用的Web服务器、应用服务器或者集成遗留系统等技术因素外，还应该考虑Web应用将用于什么样的组织框架。使用灵活的多层架构，支持多媒体内容，集成已有数据库和应用程序等，都成为Web应用架构成败的关键因素。

本章介绍Web应用的架构特性，已有的一些架构模式和框架在开发Web应用中的应用，以及一些典型的架构及其主要组件。

5.1 Web应用架构及其特性

根据Jacobson的观点，架构受系统的不同利益相关者的功能需求、性能、可重用性和可扩充性等各种质量需求，开发团队所具有的架构、模式、项目管理等经验，以及操作系统、中间件、遗留系统等开发技术等诸多因素的影响。其中，最主要的影响因素是功能需求和质量需求。需求在不断变化，因此，架构通常采用迭代开发方式进行设计，以使需求和约束的不安全性风险可预测和可控制。

然而，开发Web应用时，迭代的方式并不能满足所有架构设计问题，如集成遗留系统。必须考虑大量的功能和非功能性需求和约束，而且从技术的角度，要考虑是否已有类似的应用，如何将已有的架构进行扩展和适应性改变加以运用。设计模式和框架可以很好地支持架构决策。

5.1.1 模式

设计模式描述在特定设计环境中反复出现的设计问题并给出相应的解决方案。解决方案描述特定问题中组件及其职责、相互联系和相互作用。设计模式使人们更加简单方便地

重用已经证明了的成功的设计和架构以开发出高质量的系统。

设计模式可以从如下三个不同的抽象层次进行描述。

(1) 架构模式(Architectural Pattern)。描述系统的基本结构方案，是非常高层的结构模式，也称为概念模式，包括子系统架构及其职责、关系和相互作用。例如，层次、管道-过滤器、黑板、代理、MVC(Model View Controller)、展示-抽象-控制、反射等架构模式，其中MVC模式是非常典型的架构模式，在Web应用架构中广泛使用。

(2) 设计模式(Design Pattern)。描述特定环境下，解决通用设计问题的组件结构、关系和相互作用。"模式描述在我们周围不断重复发生的问题，以及该问题的解决方案的核心，这样，就能一次又一次地使用该方案而不必重复劳动"(Christopher Alexander在*A Pattern Language*一书中所描述)。例如，整体-部分模式、主从模式、命令模式等。

(3) 编程模式(Idiom)。描述在程序设计语言中某些功能的特定实现方式，或者利用程序设计语言的特性来实现组件内部要素之间的通信功能，与程序设计语言相关。如C++中存储管理的Counted-Pointer模式。

模式展现了特定领域中问题的成功解决方案中的静态、动态结构和相互之间的协作关系。除了低层次模式，其他模式与开发语言无关，但是它建立在一定的环境基础上。模式有助于提高系统的质量，如重用性、扩展性、性能和可维护性等。本章主要关注架构模式：MVC模式和eBusiness应用模式。

1) MVC模式

MVC(模型-视图-控制器)是一种目前广泛流行的Web应用架构模式，是一种设计思想。无论选择哪种开发语言，使用哪种开发框架，无论Web应用有多么复杂，MVC都能为理解分析应用模型提供最基本的分析方法，为构造产品提供清晰的设计框架，为Web工程提供规范的依据。

MVC的核心思想是将应用的数据和业务逻辑、控制以及页面展现分离。MVC最先被用来开发GUI领域中传统的输入、处理、输出问题。用户输入、外部世界的模型给用户的视觉反馈被模型、视图、控制器对象分离和处理。模型管理数据元素，响应状态查询和改变状态指令，是MVC最主要的核心。控制器解释鼠标和键盘输入，并将这些用户操作的指令发送给模型和/或视图来做出适当的改变。视图是用户看到并与之交互的界面。

实践证明，MVC是适合于Web应用的架构模式。MVC将Web应用的数据处理和业务逻辑、控制、视图分离，各自处理自己的任务，使得Web应用易于维护。

模型表示企业数据和业务逻辑，业务流程的处理过程对其他层来说是黑箱操作，模型接受视图请求数据，并返回最终的处理结果。视图根据客户类型显示信息，显示商业逻辑(模型)的结构，而不关心信息如何获得何时获得。控制器接受用户的输入并调用模型和视图以完成用户的需求。

MVC的处理过程是：对于每一个用户输入的请求，首先被控制器接收，并决定由哪个模型来进行处理，然后模型通过业务逻辑层处理用户的请求并返回数据，最后控制器用相应的视图格式化模型返回的数据，并通过显示页面呈现给用户。实现基于MVC的Web应用需要完成以下工作：分析应用问题，对系统进行分离；设计和实现每个视图；设计和实现每个控制器；使用可安装和卸载的控制器。

MVC的三个部分相对独立，且又能协同工作，可以降低模块之间的耦合，提高应用的

可扩展性和可维护性，加快开发和部署周期并降低开发成本。因此在构建Web应用中具有非常显著的优势，可适用于多用户的、可扩展的、可维护的、具有很高交互性的系统，如电子商务平台、CRM（Customer Relationship Management，客户关系管理）系统和ERP（Enterprise Resource Planning，企业资源计划）系统等。它可以很好地表达用户与系统的交互模式以及整个系统的程序架构模式。随着Web应用技术的不断发展与进步，现在需要通过越来越多的方式来访问Web应用。MVC模式允许用户使用各种不同样式的视图来访问同一个服务器端的代码，它包括任何Web浏览器或者无线浏览器。比如，用户可以通过计算机也可通过手机、车载设备等访问Web应用来订购某样产品，虽然订购的方式可能不一样，但处理订购产品的方式是一样的。由于模型返回的数据没有进行格式化，所以同样的组件能被不同的界面使用。例如，很多数据可能用HTML来表示，但是也有可能用WAP来表示，而这些表示所需要的仅仅是改变视图层的实现方式，而控制层和模型层无须做任何改变。在Web应用开发过程中，可以更好地分工，更好地协作，这有利于开发出高质量的Web应用。

目前在Web应用架构中流行的采用MVC模式的架构包括Java EE（及其早期版本J2EE）、.NET等。在Java EE中，模型采用EJB或JavaBean实现，控制器由Servlet实现，视图采用JSP结合HTML、XHTML、XML/XSL等Web技术实现。

2）eBusiness应用模式

eBusiness中采用如下架构模式：应用服务器（Application Server）模式给出了eBusiness应用所处的位置；服务器端会话（Server-side Session）模式将eBusiness应用的状态与Web应用的以服务器为中心的模型相结合；垂直分片（Vertical Slice）模式描述应用的组织层次之间共享并发会话的方式；服务器集群故障转移（Fail-over Through Server Clustering）模式和任务分配（Job Draining）模式是两种改进eBusiness应用可靠性的方式；心跳（Heartbeat）模式用于跟踪集群中各主机的状态；Web监视器（Web Inspector）模式用于采用Web技术显示和管理eBusiness应用；业务上下文感知目标检索（Business Context-Aware Object Retrieval）模式使用浏览器时优化交互并访问目标；预制业务对象（Prefabricated Business Objects）模式展示如何使用可重用构件组装业务。

5.1.2 框架

框架（Framework）是一组组件的综合，这些组件相互协作，为一类相关应用提供了可重用的框架结构，描述了另一种重用架构知识的方式，支持细节设计和代码重用。如MMC、MS Script Engine、Spring、Struts和Hibernate等。框架可以进行实例化，部分完成系统或者子系统，并提供了构建系统的基本构建块，还为实现特定功能定义了适应点。在面向对象系统中，框架通常由抽象类和具体类组成。

由于Java EE的标准开放，适用于Java EE的框架非常多，如Struts、Webwork、Spring、Tapestry和JSF等。这些框架的层次分割能力较好，实现了基于MVC模式的分割，而且进一步实现了一些辅助类图。

Struts框架是MVC模式的体现。其中视图主要由JSP建立，Struts本身包含了一组可扩展的自定义标签库，简化了用户页面的创建过程；模型主要由ActionForm Bean来体现；控制器主要是ActionServlet来实现，对于业务逻辑的操作通过Action、ActionMapping

和 ActionForward 来协调完成，其中 Action 是真正业务逻辑的实现者，ActionMapping 和 ActionForward 则指定不同业务逻辑或处理流程的方向。

Spring 框架主要是为了降低企业应用开发的复杂性，通过控制反转(IoC)和面向方面(AOP)的容器框架，使用基本的 JavaBeans 来完成 EJB 可以完成的事情。可以与 Struts 结合使用，也可以只使用 Hibernate 集成代码或者 JDBC 抽象层。

框架中包含的架构知识，可以在应用系统中完全采用。但是，简单重用框架的这种架构和功能也需要考虑可能带来的负面影响，如框架的使用需要专业知识，而且不同框架之间并没有集成标准，甚至开发出的应用对开发商过于依赖，等等。

5.1.3 架构分类

Web 应用的架构可以从层次特性或数据特性两个方面加以考虑。层次是为实现系统"分而治之"的原则而将系统组织成多层。层次架构是将应用根据逻辑划分成不同的部分，然后单独考虑并单独编码实现，每一部分实现不同的功能，并可能分布在不同的主机、不同的操作系统上，通常以层(Layer)来描述划分的领域，所以称为层次架构。如 Java EE 架构、企业应用集成(EAI)等采用层次架构。根据数据的结构化或非结构化特性，结构化数据基于定义好的模式(Scheme)采用如关系数据库中的表，或者 XML 文档进行描述。类似音频、视频等多媒体内容的非结构化数据，其架构单独进行设计，在 5.6 节描述。

分布式软件系统的不断增加，使得架构也相应地强调数据和消息的分布。基于分布的特点，可以将架构分为如下几类。这几类架构并不只是局限于 Web 应用。

(1) 分布对象中间件(DOM)。基于远程过程调用(RPC)机制，允许透明地访问远程对象。如微软的 DCOM 和 Sun 的 EJB，还有些 DOM 系统还能够支持不同平台上的对象交互，如 CORBA。

(2) 虚共享内存(VSM)。VSM 模型支持分布进程访问同样的数据。进程自身访问共享内存，中间件对于进程而言是透明的，用于分发数据。这些数据可以在系统的任何地方，因此被冠以"虚"，如 Corso 和 Equip。

(3) 面向消息中间件(MOM)。主要提供异步消息交互功能。异步通信和同步的主要不同在于，消息发送给接收者，而不关心其状态，如接收者在消息发送时不在线，MOM 却确保消息仍然发送。典型的 MOM 系统如 Sun 的 JMS(Java 消息服务)和微软的 MSMQ 等。

(4) P2P。两台设备(对等点)可以直接通信，而无须服务器，这些对等点之间是平等的。这类系统如 JXTA 和 Xmiddle。

(5) 面向服务的中间件(SOM)。这类中间件在 DOM 系统的基础上，加上服务的概念，尤其是 Web 服务。这里，服务是指一组对象及其行为。这些对象使用定义好的接口，使其可以被其他系统或服务使用。SOM 定义服务之间的通信协议，提供位置和迁移透明的服务访问，因此可以支持在平台范围之外的服务集成。如 Sun 的 Jini 系统。

5.1.4 Web 应用架构特性

开发 Web 应用很重要的一个需求是质量，比传统软件有更高的要求，尤其是考虑可变性、性能、安全性、扩充性和可靠性等，要求特殊的针对 Web 应用运行和维护的技术基础。

因此，我们分 Web 基础架构和 Web 应用架构两种加以描述。Web 基础架构指 Web 平台架构（Web Platform Architecture，WPA），Web 应用架构（Web Application Architecture，WAA）强调 Web 应用所处的问题域。本章中以平台架构为主。

WPA 面向很多问题。应用服务器如 Java EE、.NET 平台提供会话处理、协议包装、数据访问等基础服务。除了应用服务器外，有专门针对安全（如防火墙）、性能（缓存代理）或数据集成等方面的特定架构（如 EAI）。使用众多不同系统使得更难评估和维护各种质量需求，如使用的组件和第三方产品越来越多，使得性能需求越来越难以满足。

另外，开发 Web 应用还面临着技术架构的异构性和不成熟性，应用服务器的版本快速更新，导致其性能和兼容性下降。撇开异构和不成熟问题，目前，为了解决问题，Web 应用使用很多不同的技术基础架构，从开源框架（如 Structs、Spring、Hibernate 和 Cocoon）到应用服务器（如 Jboss），再到门户框架（如 Brazil 和 JetSpeed 等），甚至同时使用多个框架。

Web 应用架构的另一个显著特性是其国际化，要求 WPA 支持不同的语言、字符集和展现机制，如从右向左显示阿拉伯文。如 Java 平台支持国际化机制。

在设计 Web 应用架构时，必须考虑所有上述这些方面，尤其是 WPA 要提供解决共有问题的功能，并且定义 Web 应用变化的环境。通常，Web 应用架构包括如下通用组件，如图 5.1 所示。

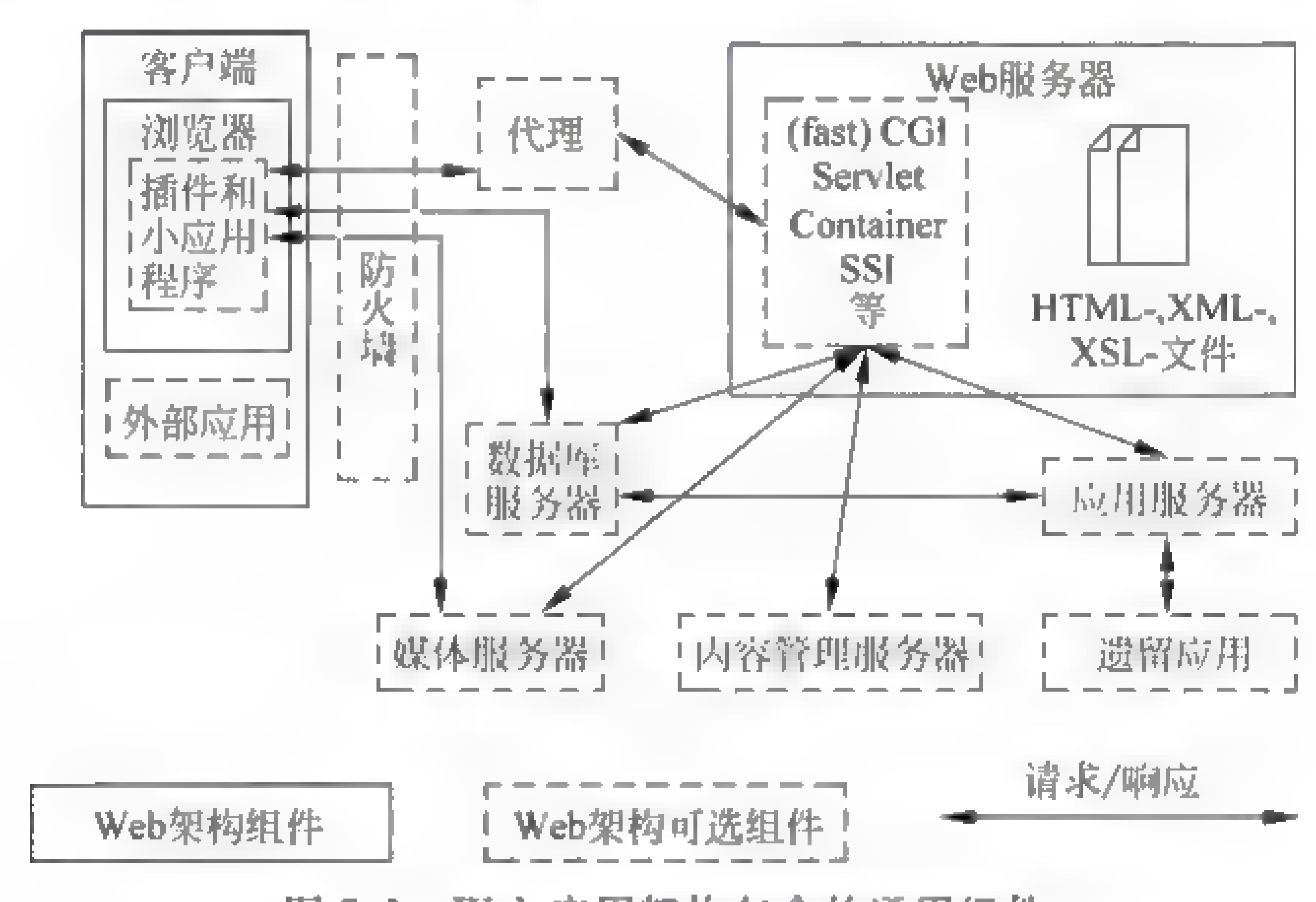


图 5.1 Web 应用架构包含的通用组件

(1) 客户端。通常是浏览器，用户通过浏览器控制和 Web 应用的交互，也可以在浏览器的基础上，扩展安装一些插件或小应用程序。

(2) 防火墙。将服务器和互联网之间的连接进行过滤的软件。

(3) 代理。通常用于 Web 页面的缓存，或者根据用户或者角色的不同提供个性化内容，或者用于用户跟踪。

(4) Web 服务器。支持 HTTP、HTTPS 等各种 Web 协议的软件，处理客户请求。

(5) 数据库服务器。主要指关系型数据库，其中存储结构化数据，如表。

(6) 媒体服务器。用于非结构数据流内容的组件，如音频或视频。

(7) 内容管理服务器。类似于数据库服务器，存放 Web 应用所需的内容，一般是类似于 XML 文档的半结构化数据。

(8) 应用服务器。用于存放和支持由多个应用所需的功能，如工作流或个性化。

(9) 遗留应用。需要作为内容组件或者外部组件集成在Web应用中的已有旧应用。

这些组件及其交互通常基于请求-响应模式，由浏览器给其他组件（如Web服务器）发出请求，经过相应处理之后，响应也通过同样的渠道返回给浏览器（同步通信方式）。本章从模型驱动、层次、集成、数据方面分别介绍Web应用架构及其包含的组件的组织形式。

5.2 模型驱动架构

模型驱动架构（Model Driven Architecture，MDA）是由OMG（Object Management Group，对象管理组织）于2001年提出来的模型驱动开发的概念框架，由统一建模语言（UML）、元对象机制（Meta Object Facility，MOF）、可扩展标记语言（XML）、公共仓库元模型（Common Warehouse Metamodel，CWM）、OCL（Object Constraint Language，对象约束语言）和ASL（Action Specification Language，动作规约语言）等支持。其主要思想是将业务功能的分析设计与实现技术和平台之间的紧耦合关系相分离，从而将技术与平台变化对系统的影响降低到最小程度，适用于设计、部署、集成等软件开发的整个生命周期。

MDA是一个开放的独立于软件供应商的架构，支持众多应用领域和技术平台。MDA将开发行为提升到描述业务需求的功能模型（Platform Independent Model，PIM，平台无关模型），将PIM抽象出来而不涉及技术细节，用OCL的前置、后置条件和动作语言进行描述，然后针对不同实现技术与平台采用OMG的标准映射（Mapping），通过这些映射及辅助工具将PIM转换成与具体实现技术和平台相关的应用模型（Platform Specific Model，PSM，平台相关模型），如Web服务、.NET、Java EE和CORBA等，再将PSM不断求精直至形成各种平台支持的最后代码，如XML、Java、C++等，如图5.2所示。这样，当业务运行稳定不变，而因为技术变化等原因导致平台升级等时，PIM就不需要变化，只要通过平台相关的模型映射，重新生成PSM，极大地加强了应用模型与领域模型在整个软件生命周期中的复用。

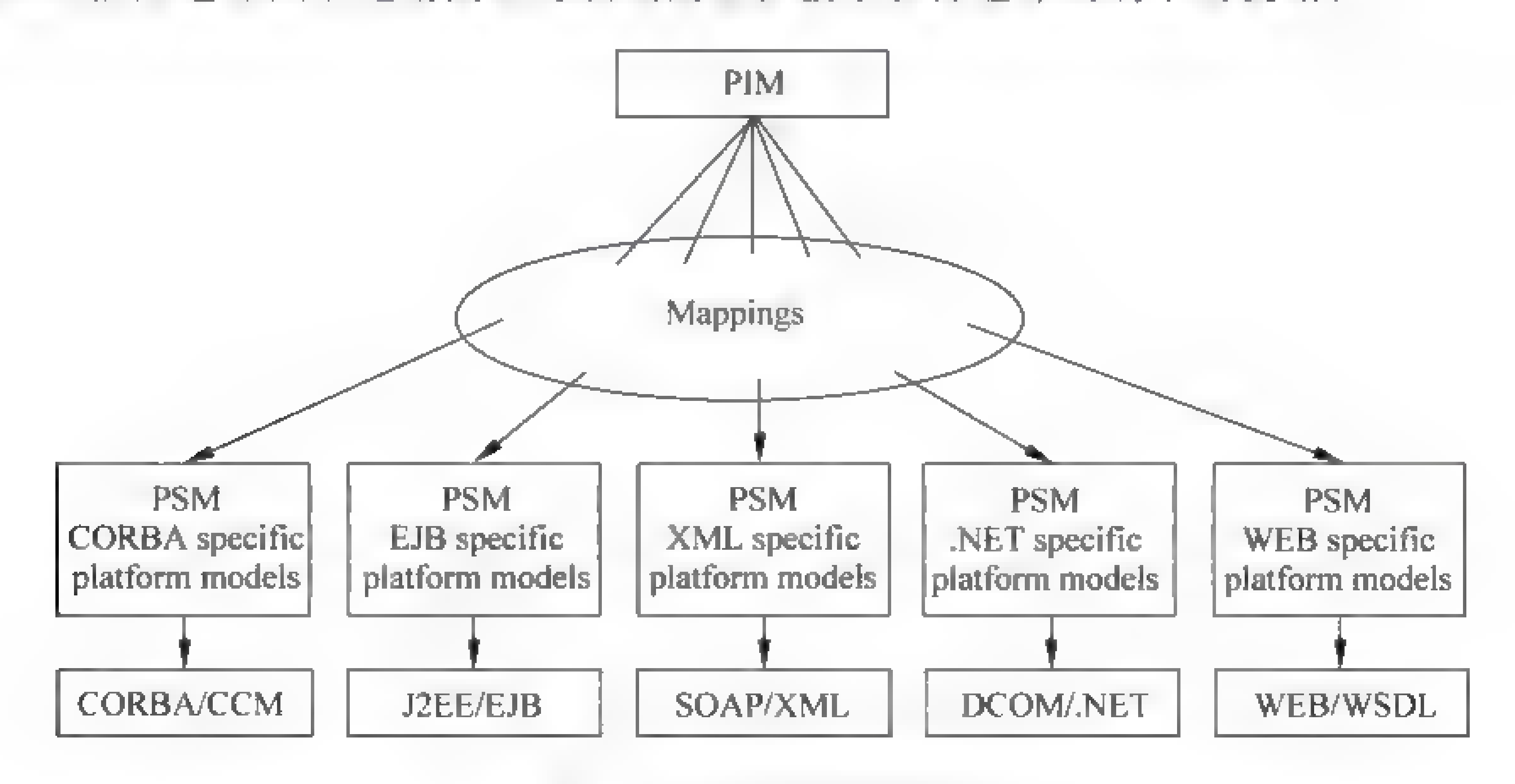


图5.2 MDA的主要思想

OMG在MDA中采用多种中间件平台，也是集成的有效架构。映射和转换是MDA的两个重要机制，在转换过程中，需要进行规范映射和转换定义。映射规范描述了PIM的元模型和PSM元模型之间的对应性，转换定义是对模型转换时具体使用的转换语言的描述。

MDA的主要优势主要表现为如下几个方面。

(1) 提高生产效率。模型驱动开发使开发人员的焦点转移到了PIM的设计和开发上。所需的PSM是通过转换自动从PIM生成的，目标平台的细节通过从PIM的转换自动加入PSM。转换只需要被定义一次，然后就可以在开发中多次应用这个转换。此外，模型的可读性和可维护性较强，易于修改和完善。因此提高了生产效率。

(2) 提高了可重用性。当中间件技术改变时，可以直接使用已经过测试并投入使用的平台无关模型，开发人员只需要重新开发模型转换工具，替换原有的平台相关模型即可。

(3) 增强可移植性。可移植性是通过把开发焦点转移到PIM而获得的。因为PIM是跨平台的、与平台无关的。同一个PIM可以被自动转换成多个不同平台的PSM。因此在这个意义上，PIM层次上定义的所有东西都是完全可移植的。

(4) 支持互操作性。如图5.3所示，从一个PIM生成的多个PSM之间可能会有联系。这种联系被称为桥接器。不同平台间的PSM不能直接联系。需要把一个平台的概念转换到另一个平台，这称为互操作性。因此还需要生成PSM之间的桥接器。桥接器简化了应用模型和创建集成应用，这种集成可以是企业内部应用，也可以是跨不同企业的应用。

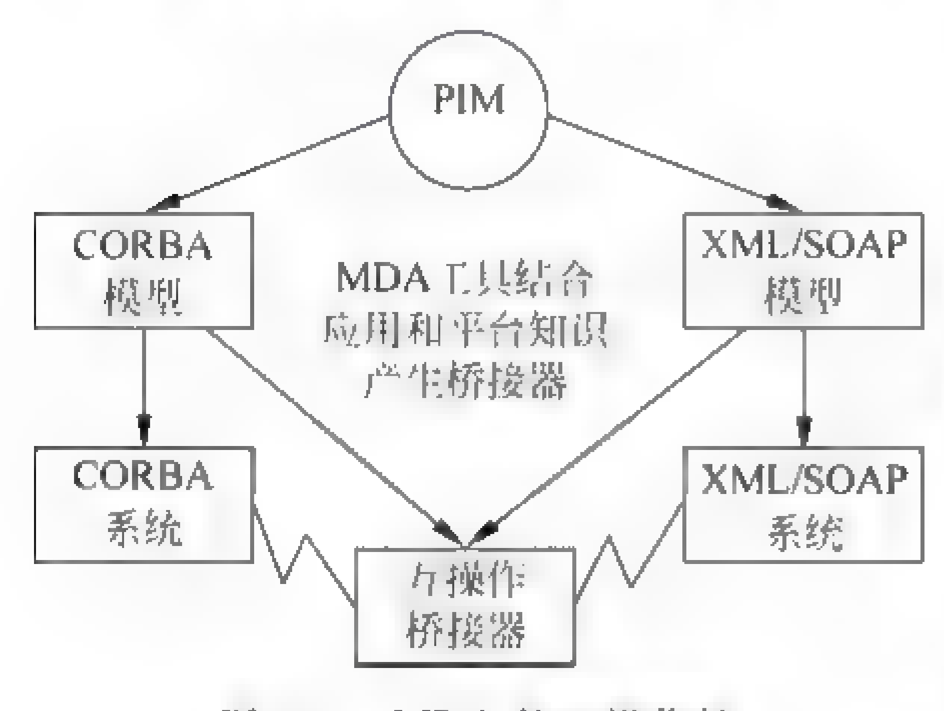


图5.3 MDA的互操作性

(5) 提高了系统的可验证性。MDA将模型的构建、模型与模型的转换、模型与代码间的转换都统一到一致的技术框架下，提供了一致的解决方案，实现了其标准化和自动化。这使开发过程的各个阶段的工作具有较好的一致性和连贯性，各阶段的工作可以相互支持，逆向验证。

(6) 便于维护。模型驱动开发工程中，开发者可以把注意力放在PIM上。PIM抽象层次比代码高，而模型是代码的精确表现，因此PIM起到了软件系统所需要的高层次文档的作用，便于系统的维护。

上述MDA的诸多优点也使得MDA在各行各业中应用越来越广，支持工具也越来越多。工业界把UML作为标准加以运用，IBM和Borland等软件开发商的支持，为MDA的推广提供了坚实的基础。从工具对MDA开发过程的支持程度来划分，可以将其分为两类：完全式和不完全式。①完全式，即完全支持整个开发过程，从UML建模，模型转换到代码生成。比较有代表性的有Metanology Corporation的Meta Development Environment (MDE)，Interactive Object Software的Arcstyler，Compuware Corp的OptimalJ以及IBM的Rational XDE。由于对MDA开发过程的完全支持，完全式是现在应用最为广泛的，也是发展最快的。②不完全式，即支持将从其他模型工具得到的模型生成代码，比较有代表性的有Codagen Technology Corp的Architect以及Telelogic的Tau。

5.3 层次架构

Web应用的用户需求在不断变更，但通常情况下不会整个系统都改变，变化的只是系统的一部分功能或一部分逻辑而已。将展示逻辑、业务逻辑和数据访问逻辑分离，能够将变

化局域化在某一领域，局限于这一层。如展示层不同页面组合或功能组合不会涉及业务逻辑处理组件的变化，对数据存储和提取的优化也不会连带其他层的程序变更，降低部分修改对系统整体的影响，从而有助于提高系统的可扩展性和可维护性。层次架构更容易容纳新的技术和变化，容许任何一层变更所使用的技术。例如业务逻辑层既可以通过以 DLL 动态链接库形式提供组件调用，也可以通过 Web 服务接口实现跨平台的服务调用。

5.3.1　两层架构

两层(2 Tier)架构也称为客户/服务器架构，应用逻辑分布在服务器端，向客户端提供服务。如图 5.4 所示，Web 应用的两层架构由客户端和服务器端组成，客户端向服务器端发出请求，其请求直接指向静态 HTML 页面，或者通过 Web 服务器和业务逻辑访问数据库，或者动态 HTML 页面，来响应客户端请求并给客户端返回结果。

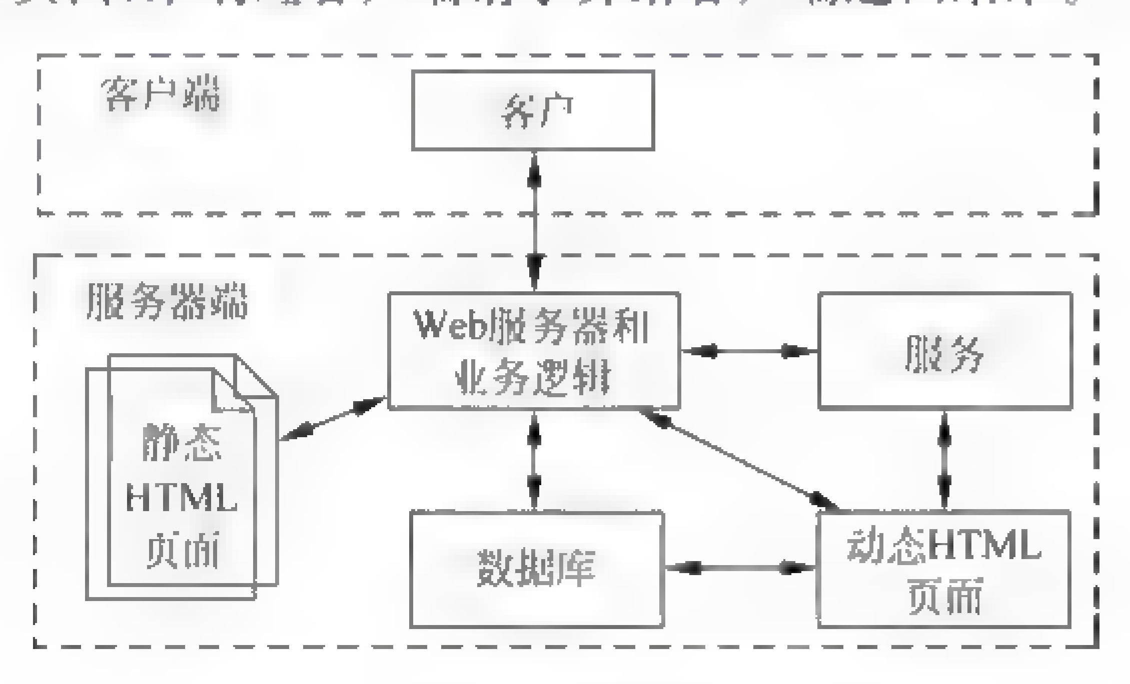


图 5.4　Web 应用的两层架构

两层架构非常适用于简单的 Web 应用。然而，当并发用户数量增多时，服务器端的性能会因为负载过重而大大衰减；Web 服务器和应用逻辑没有分离，当 Web 应用结构复杂性上升时，Web 应用的可维护性和可扩展性将受到极大的制约。

相比而言，三层架构和多层架构更适合具有大量并发客户或者需要访问遗留系统来提供复杂业务处理等情况。

5.3.2　三层架构

三层(3-Tier)架构是将 Web 应用中组件划分为展示层、业务逻辑层(应用服务层)、数据层，如图 5.5 所示。展示层封装与用户或其他系统的交互，如浏览器，执行展示逻辑；业务逻辑层组成业务逻辑，在应用服务器上执行业务逻辑；数据层封装对持久存储的操作。

展示层
业务逻辑层
数据层

图 5.5　三层架构

1）展示层

展示层是客户端服务程序，提供系统的用户接口和各种操作界面，包括数据输入和结果显示，向业务逻辑层请求调用核心业务逻辑服务。

2）业务逻辑层

业务逻辑层无疑是系统架构中体现核心价值的部分。它的关注点主要集中在业务逻辑的制定、业务流程的实现等与业务需求有关的系统设计，也就是说它是与系统所应对的领域逻辑有关，很多时候，也将业务逻辑层称为领域层。Martin Fowler 在 *Patterns of*

Enterprise Application Architecture(《企业应用架构的模式》)一书中，将架构分为三层：展示层、领域层和数据源层。业务逻辑层的设计对于一个支持可扩展的架构尤为关键，因为它扮演了两个不同的角色。对于数据访问层而言，它是调用者；对于展示层而言，它却是被调用者。依赖与被依赖的关系都纠结在业务逻辑层上，如何实现依赖关系的解耦，则是除了实现业务逻辑之外留给设计人员的任务。

3）数据层

数据访问层有时候也称为持久层，其功能主要是负责持久存储数据的访问，可以访问数据库系统、二进制文件、文本文档或是 XML 文档。简单的说法就是实现对数据表的选择、插入、更新和修改的操作。

相比两层架构，三层架构的优点主要表现在以下几个方面：层间依赖性降低，易于实现层的分工、替换和各层逻辑的重用，也有利于标准化，有利于后期的维护和升级。

三层架构同时也有一些缺点。一定程度上降低了系统的性能，如对数据库必须通过中间层来完成；有时会导致级联的修改，如在展示层中需要增加一个功能，为保证其设计符合分层结构，可能需要在相应的业务逻辑层和数据访问层中都增加相应的代码。基于三层的 Web 应用在层间通信时会带来性能损耗，复杂的配置管理会降低系统的负载能力。

尽管三层架构也具有诸多缺点，但是对于大型 Web 应用而言，其优点更加明显，也因此在实际的 Web 应用开发中，尤其在大型 Web 应用开发中，使用得越来越广。

一种典型的三层 Web 应用架构是 ASP. NET 应用架构，如图 5.6 所示，每层又包含一些子层。将这些子层再重新划分成层，即可形成多层架构。

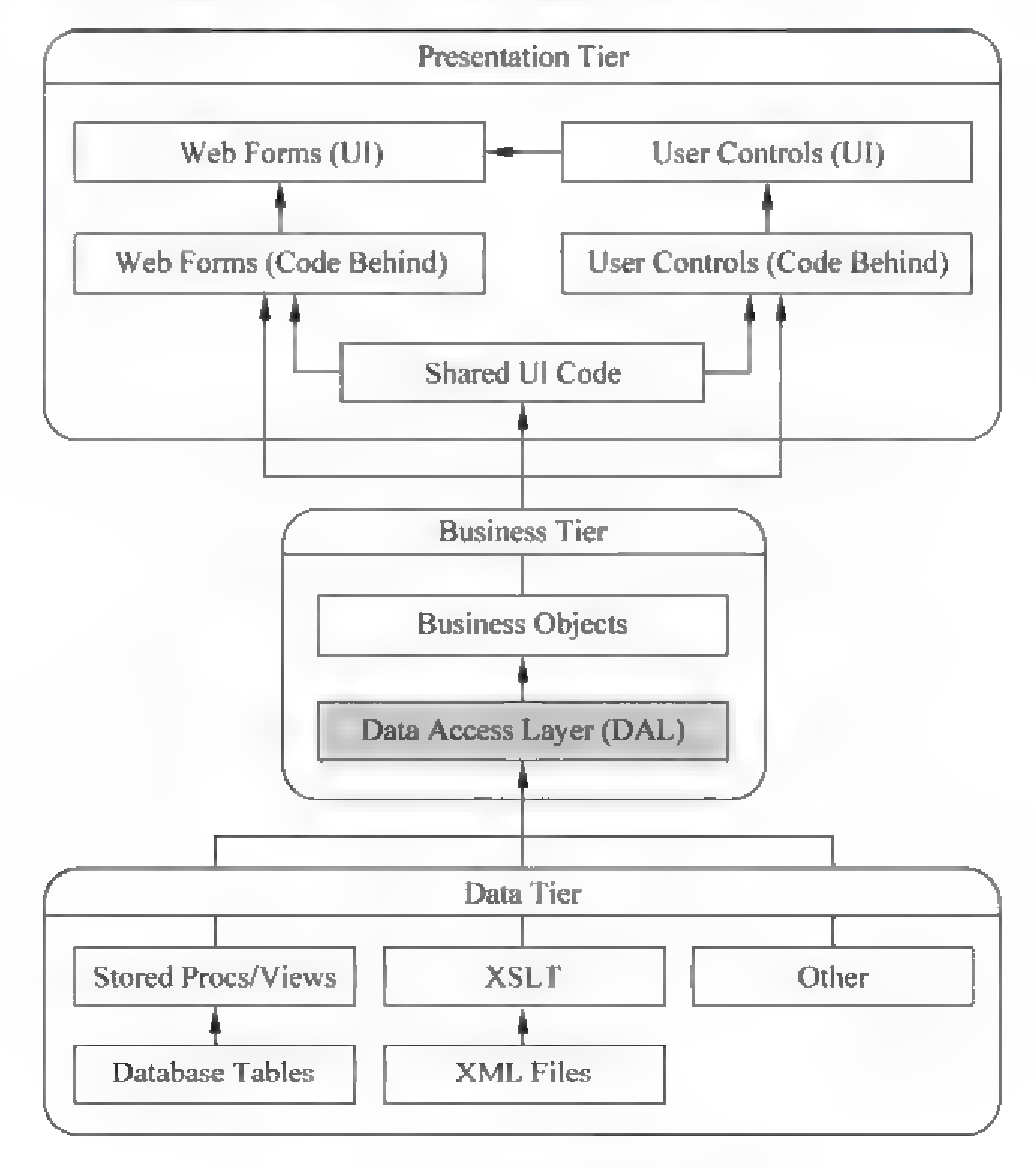


图 5.6 .NET 应用三层架构

5.3.3 N层架构

领域驱动设计的先驱 Eric Evans 对业务逻辑层作了更细致的划分，细分为应用层与领域层，通过分层进一步将应用逻辑与领域逻辑相分离。

N层(N Tier，指三层以上的多层)架构是在三层架构的基础上，将服务集成于应用服务器，展示逻辑由 Web 服务器完成，和数据层的交互由数据访问层负责，客户端负责页面展示，另外还有安全(如防火墙)和个性化(如代理)等层，如图 5.7 所示。它是一种能够满足企业级信息共享、业务操作的应用架构。其主要特点是各种服务采用规定的接口，包含在应用服务器中。如 WebSphere 应用服务器包含 WebSphere 业务组件，并且具有应用服务器的分布和负载均衡能力。

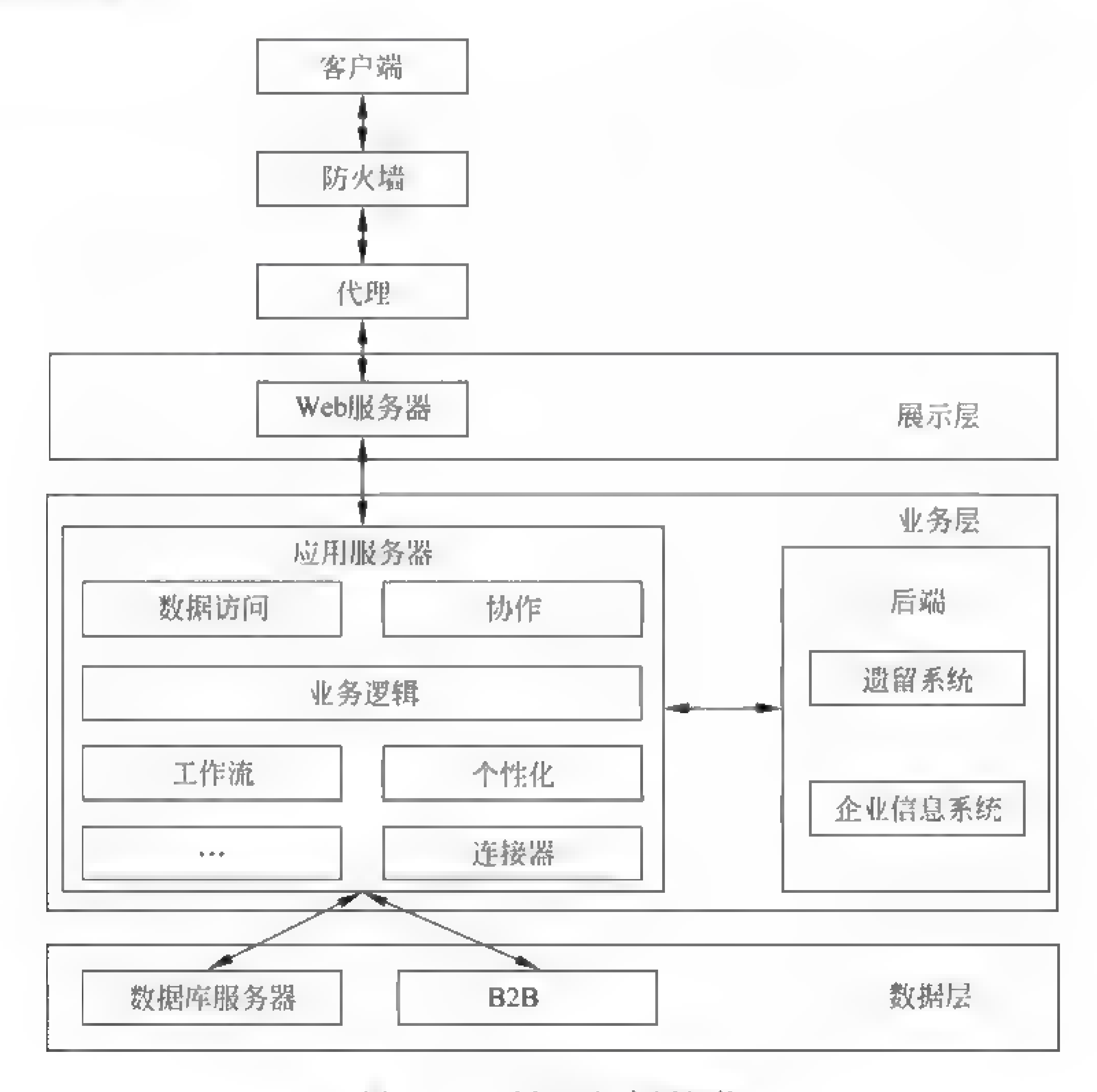


图 5.7 N层 Web 应用架构

另一种典型的层次架构是从上到下分为5层：展示层、展示逻辑层、业务层、数据访问层和数据层。其中只有展示层处于客户机上，通过 Web 页面技术进行展示，其他层都处于服务器端和(或)数据库服务器端。

N层架构的核心是提供可规模化特性，一方面是从服务负载上可规模化，能同时为极大规模的用户提供服务；另一方面是服务功能上的可规模化，可形成极大规模的 Web 应用集群系统，各分系统可以共享信息、服务，形成企业级的信息高速公路。N层可以分别放在各自不同的硬件系统上，所以灵活性很高，能够适应客户机数目的增加和处理负荷的变动。例如，在追加新的业务处理时，可以相应增加装载功能层的服务器。因此，系统规模越大这

种形态的优点就越显著。

使用N层架构，每一层都可以在仅仅更改很少量的代码后，就能放到物理上不同的服务器上使用，因此结构灵活而且性能更佳。此外，层次之间独立性更高，因此层次之间的更改、更新更加独立。例如，如果把数据访问代码与业务逻辑层分离，当数据库服务器更改后，只需要更改数据访问的代码，因为业务逻辑层是不变的，因此无需要更改业务逻辑层。这一特性对于大型Web应用而言尤为重要。

Java EE架构是目前最常用的N层架构之一，还很好地实现了MVC模式。一种典型的Java EE架构如图5.8所示。实现MVC模式的还有JSP Model 2、JSP Model 2在Structs中的实现、OOHDM-Java2等。

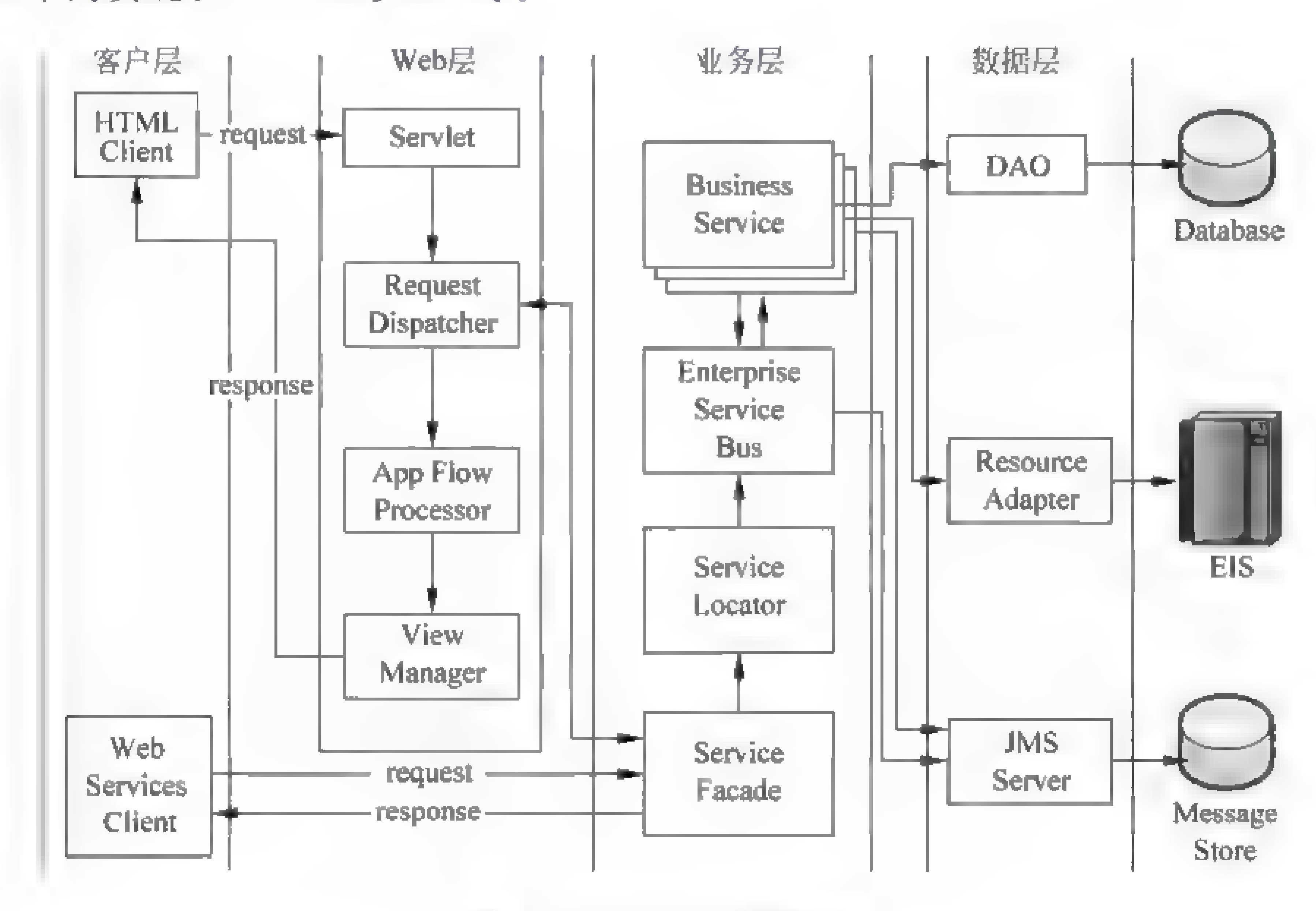


图5.8 Java EE应用架构

越来越多的商业应用服务器对数据内容的处理进行了优化，对多媒体内容和超文本结构的支持或优化却还远远不够。目前，出现了如Informix Video Foundation DataBlade提供了开放且可扩充的架构，开发者可以在数据库中使用视频技术；WebRatio建模工具能够将超文本映射到Java EE和.NET。说明了大多扩充支持是基于Java EE等架构之上进行的。

5.4 集成架构

随着网络和信息化进程的快速发展，Web应用需要包括和外部系统或内部系统的集成，如与外部商业伙伴的已经存在的应用、已经存在的数据库和接口等进行集成。可以通过在展示层面、应用逻辑层面、内容层面3个层面将这些已有系统集成到Web应用中。

5.4.1 门户

门户(Portal)是指基于Web技术，并针对具体用户或社区的应用平台。如今，在Internet上随处可见如腾讯、雅虎、新浪和网易等网络信息门户，把用户在Internet上可能用到的众多内容与服务都集中为一个Web应用，体现在其主页上，使用户通过这个“大门”进入以使用所需的内容与服务。门户的主要功能是可以提供灵活整合、集成各种如OA、固定资产、E mail、HR等应用系统，并把整合好的内容个性化地展现给终端用户。用户可通过统一平台使用不同服务和操作不同应用，并进行协同工作与社区服务。

门户是展示层面的集成，集成多样化内容服务的Web应用，如图5.9所示。展示集成是最简单的集成方式之一，将不同服务商提供的分布在不同服务器结点上的内容和应用逻辑，根据被称为Portlet的门户组件，通过页面集成作为适合门户导航结构和布局的一部分，由独立的聚合组件将不同的门户片段集成在一起，形成一个新的、视觉统一的Web展示页面。这种聚合可以由门户提供者设计，也可以由用户自定义。基于门户的Web应用有主要提供信息内容的网络信息门户，也有提供企业应用服务的企业门户。

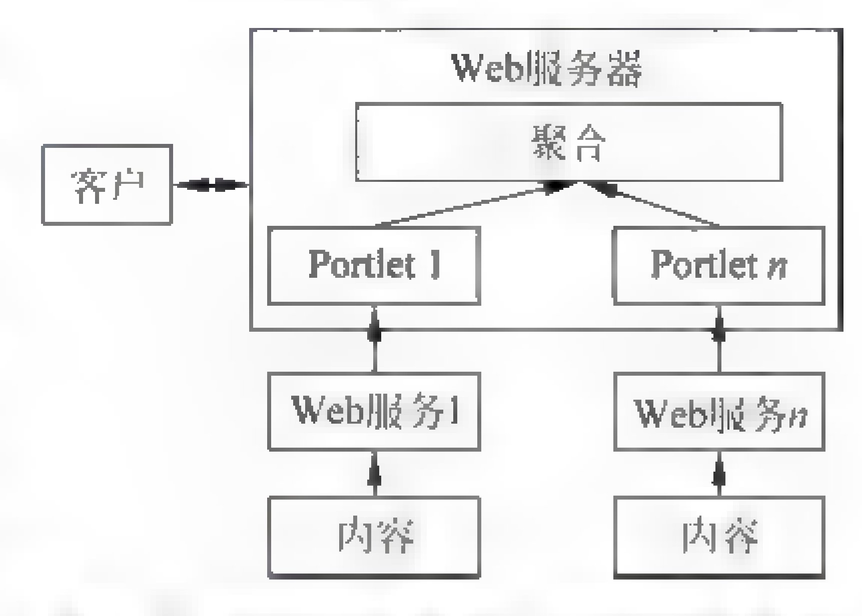


图5.9 基于门户的Web应用架构示例

门户可以分为两种类型：水平门户，包括新浪、腾讯等通用目的站点；垂直门户，如W3C联盟、语义Web社区门户等关注特定功能域的门户。门户服务器如开源项目JetSpeed。

5.4.2 EAI

EAI(Enterprise Application Integration，企业应用集成)强调内容层面和应用逻辑层面集成的架构，将不同数据源和基于各种不同平台、用不同方案建立的异构应用集成的一种方法和技术。严格来说，EAI主要的关注点在于集成遗留系统。EAI通过中间件连接企业内外各种业务相关应用以及数据源，从而满足电子商务、ERP、CRM、SCM、OA、数据库、数据仓库等重要系统之间无缝共享和交换数据的需要。

EAI起源于B2B(业务到业务)的集成，涉及技术广泛，实施复杂，它将进程、软件、标准和硬件联合起来，对两个或更多的企业应用实现无缝集成，成为一个整体。这类集成可通过P2P、数据交付、数据集成、功能交付、功能重建和移植来实现。利用EAI企业可以将核心应用和新的Internet解决方案结合起来，如图5.10所示。

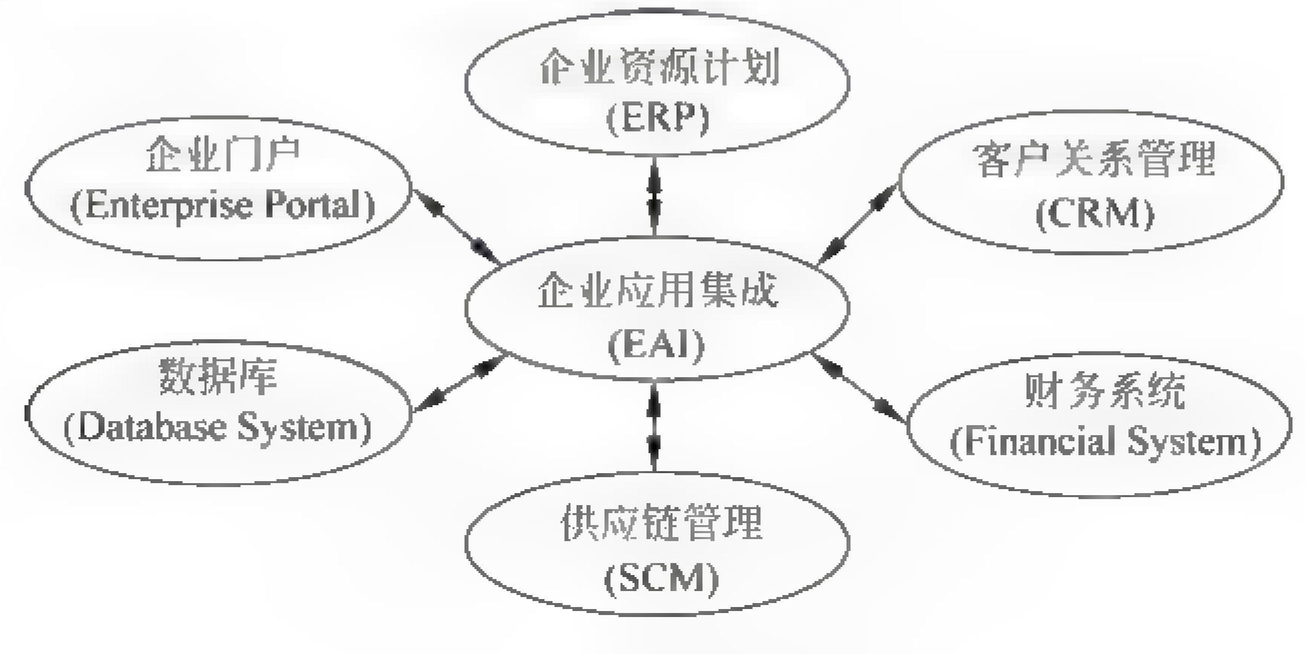


图5.10 EAI解决方案

随着应用集成研究的深入，人们提出了不同的集成模型，如图5.11所示的包装器/适配器模式，不同的模型实现不同的应用集成需求。目前，信息集成系统多采用基于包装器/中介器（Wrapper Mediator）方法。这种方式通过提供所有异构数据源（如数据库、遗留系统、Web数据源等）或系统进行集成。与信息集成类似，对遗留系统集成使用特定中间件，通常由应用服务器和XML技术相结合，如J2EE Connector架构将遗留系统集成到J2EE平台。

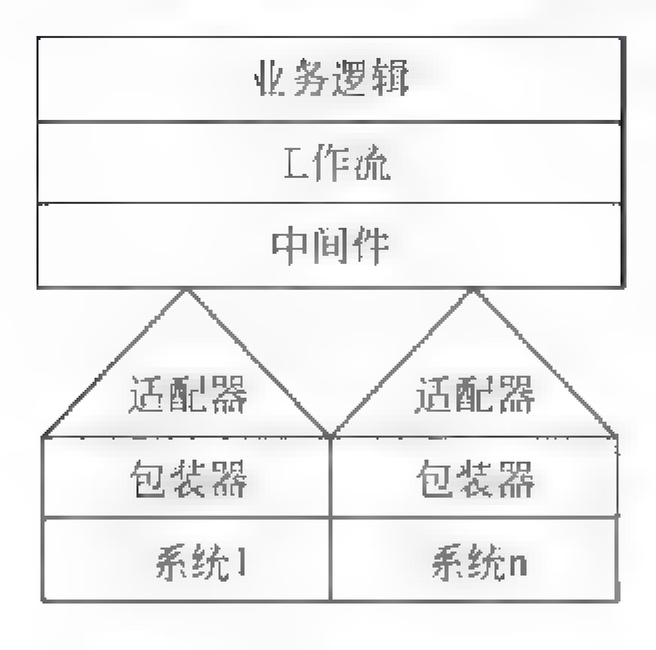


图5.11 EAI架构示例

分布式对象技术是EAI集成常用的技术，以CORBA、DCOM和RMI为代表的分布式对象技术是传统企业集成的主要技术手段，它们为实现EAI的可移植性、可扩展性和可重用性提供了解决途径。随着Web服务的发展，越来越多地将Web服务与EAI技术结合使用。基于Web服务的EAI的核心是将待集成的应用、业务功能或者工作处理包装为Web服务。

5.4.3 SOA

SOA（Service-Oriented Architecture，面向服务的架构）提供一种集成框架，其关键是“服务”的概念。它将应用程序的业务功能单元称为服务，通过这些服务之间定义好的接口和约定进行集成，形成一种架构模型，从而构成整个应用。这种集成方式与接口的具体实现方式无关，不需要考虑特定系统或实现的特征，因而更加灵活，如可以基于策略选择具有相同接口的不同服务器提供者。SOA可以进行多对多集成，如跨企业的各类客户可以通过不同方式使用或重用应用程序。SOA的层次架构如图5.12所示。服务层是SOA的基础，可以直接被业务过程调用。

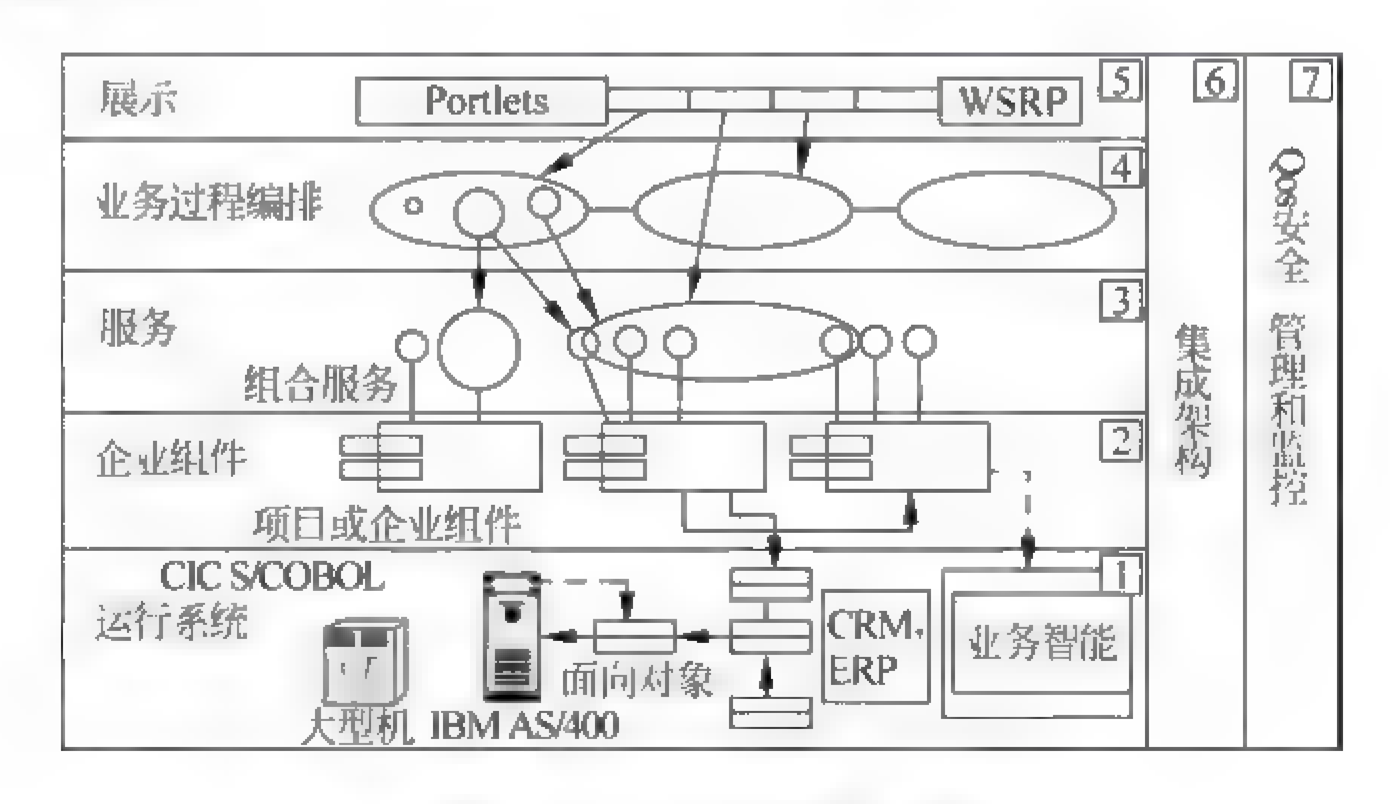


图5.12 SOA的层次架构

SOA是设计和构建松散耦合的解决方案，能够以程序化的、可访问的形式公开业务功能，通过服务接口的标准化描述使任何异构平台和任何其他应用程序可以通过已发布和可发现的接口来使用这些服务。SOA一般提供查找服务、注册服务和绑定服务三种操作。服务请求者使用查找服务来定位服务；服务提供者将服务的信息发布到服务注册者，服务的信息包括所有与该服务交互所必要的信息，如网络位置、传输协议以及消息格式等；一旦服

务请求发现合适的服务，它将根据服务描述中的信息在运行时直接激活服务。

服务请求者到服务提供者的绑定与服务之间是松耦合的，服务请求者不知道提供者实现的技术细节，如程序设计语言、部署平台等，服务请求者往往是通过消息调用操作，保护服务的独立性。实现了服务间的松散耦合，就可以通过插件(Plug-in)的方式不断向SOA架构中加入新服务或者更新已有的服务，而且不会影响到其他的服务，因此提供了更高层次的重用性。

SOA中的服务概念与Web服务是两个不同层面的问题。SOA面向商业应用，是一种概念模式，而Web服务则面向技术框架，是一种实现模式。Web服务只是实现SOA的方式之一，也可以用其他的方式来实现SOA，例如通过JMS、JDBC、.NET Remoting、CORBA等。

集成外部Web应用还面临着很多很艰巨的问题，如集成之后的系统的性能和可扩充性问题。也就是说，我们无法预计最终系统的响应时间或嵌入组件的可靠性，因此，服务质量成为集成外部服务时的重要问题。

5.5　面向数据的架构

数据能够被组织成数据库中的结构化类型、文档管理系统中使用的文档类型，或者多媒体服务器中的多媒体数据类型。从数据的角度而言，Web应用并非是这些数据类型中的某一种，而往往是集合了文档、媒体和数据库，也因此具有不同的应用架构。

5.5.1　以数据库为中心的架构

数据库技术已经非常成熟，Oracle、SQL Server、DB2、Sybase、MySQL和PostgreSQL等数据库系统的功能日趋强大，并且能够直接由Web服务器扩展访问(两层架构)或者Web应用服务器访问(三层或N层架构)。对于Web应用中集成数据库(尤其是关系数据库)而言，有很多工具和方法可用，很容易被集成。对于不同的平台而言，有应用程序接口可用，例如JDBC和ODBC可以用来访问关系型数据库。值得一提的是，针对以数据为中心的Web应用，为了提高效率，有各种优化的方式和辅助技术。对数据库连接技术有兴趣的读者请参阅相关资料。

5.5.2　Web文档管理架构

除了结构化数据和多媒体数据分别存储于数据库和媒体服务器上外，Web应用的内容还以文档的形式进行处理。内容管理架构是将来自不同数据源的文档，集成到Web应用的机制。

图5.13展示了一个内容管理架构，包括客户端、内容交付系统、内容管理服务器、其他外部资源和企业联合服务等组件。Web服务器收到客户端请求并将它传给内容交付服务器，内容交付服务器负责分发内容，也可能进行内容的缓存。如果被请求的内容不在缓存中，那么会将请求发送到内容管理服务器。不论内容作为静态文档形式还是在内容数据库中，都可以直接在内容管理服务器上访问这些资源，甚至能够从外部访问到。根据这种集成类型，可以访问外部数据库或联合服务获取到外部内容。企业联合服务与访问数据库不同，它还能够处理额外的功能，例如自动发送许可权。

在内容管理架构中，发布器可以采用集成已有框架来实现，如Cocoon2框架用于生成XML文档，并发布于Web应用中。一般情况下，Cocoon2可用于将XML内容文档转换为

不同格式的输出，其底层采用管道模型进行处理，即接收请求之后转给预定义配置文件中配置好的产生器(Generator)，然后由转换器(Transformer)进行多步转换处理生成所需格式的输出，到底分多少步以及每步具体的用途是什么，根据设计意图进行设计。在所有处理步骤中都可以访问请求的信息，如用户界面定制信息等。产生器在开始需要处理信息时调用，负责解析 XML 文档。Cocoon2 中采用 SAX 解析器，读 XML 文档，然后生成 SAX 消息，称为事件，并将这些事件转至管道中。这些 SAX 事件可以用于转发文档或者转换文档，也就是起着准备要处理文档的作用。转换器类似于 XSL 样式单。而串行器(Serializer)负责将输出转换成所需格式，如 HTML、XML 等。在 Cocoon2 中，采用 XML 配置文件来选择和定义产生器、转换器和串行器，如根据请求和处理结果要求，配置控制处理管道的过程等。

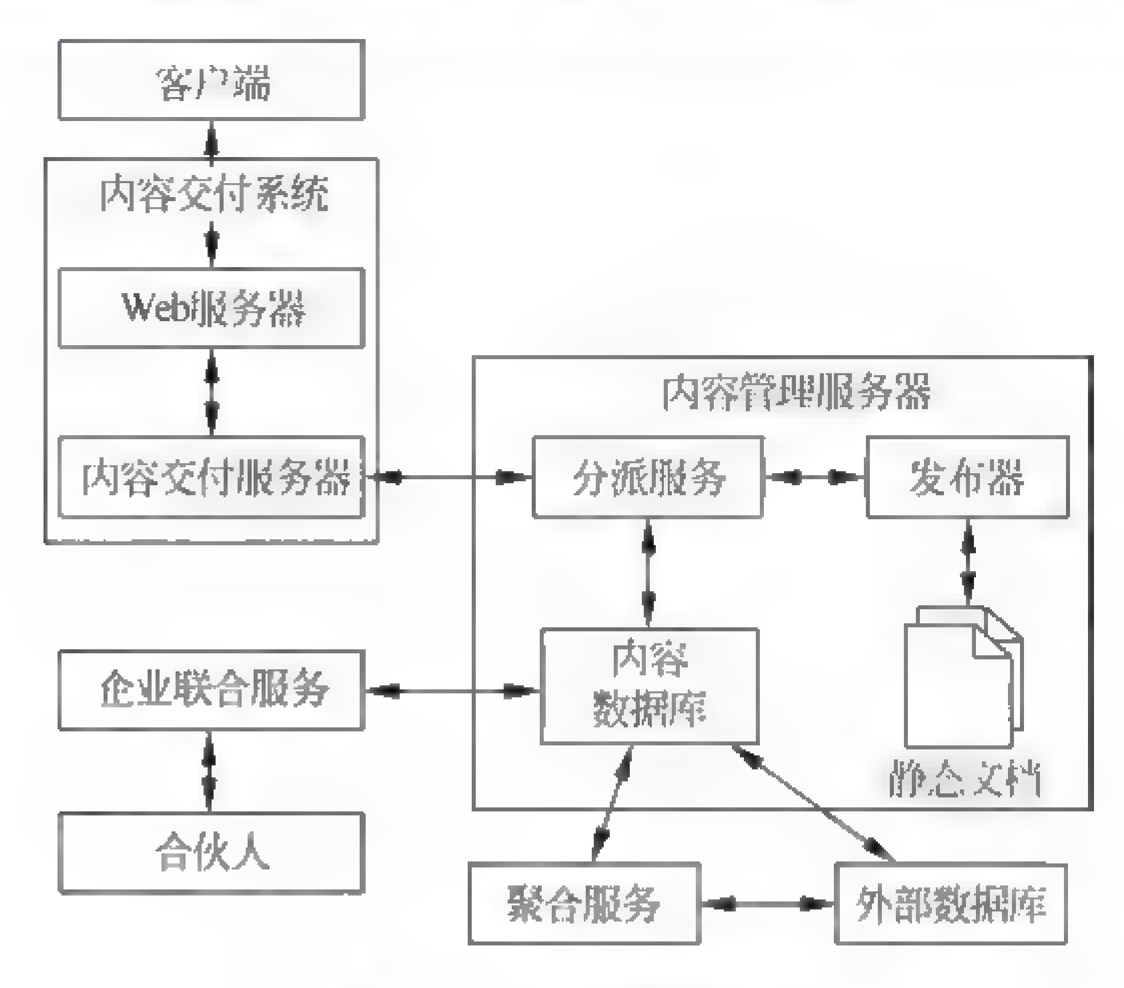


图 5.13 内容管理架构

5.5.3 流媒体数据的架构

除了一般文档内容之外，大型的 Web 应用中经常包括多媒体内容，因此，多媒体内容成为 Web 应用中重要的组成部分之一。多媒体数据量难以在 Web 应用中确定，因此，也就影响着多媒体 Web 应用的架构和设计。

多媒体数据包括视频、音频等，能够和传输其他 Web 应用数据一样，通过标准的因特网协议(如 HTTP、FTP 等)进行传输，这些多媒体内容和相应的方法在 Web 应用中大量使用。但是，用户使用 HTTP 和 FTP 等下载多媒体时，经常面临下载速度慢的问题。为了解决这一问题，使用流技术来最小化多媒体内容播放的等待时间。流表示一个客户端能够开始播放音频或视频仅仅在它从一台服务器收到所需文件之后的极短时间内。这种技术避免了在开始播放前需要下载整个文件的问题。

流媒体指在 Internet/Intranet 中使用流式传输技术的连续时间媒体，如音频、视频或多媒体文件。流式媒体在播放前并不下载整个文件，只将开始部分内容存入内存。流媒体的数据流随时传送随时播放，只是在开始时有一些延迟，当然也要考虑带宽的限制和可能造成的抖动。

实现流媒体的关键技术是流传输，主要指通过网络传送媒体(如视频、音频)的技术总称。将整个音频、视频以及 3D 等多媒体文件经过特殊的压缩方式分成一个个压缩包，由视

频服务器通过 Internet 连续、实时传送至用户计算机。在采用流传输方式的系统中，用户不必像采用下载方式那样等到整个文件全部下载完毕，而是只需经过几秒或几十秒的启动延时即可在用户的计算机上利用解压设备(硬件或软件)对压缩的多媒体文件解压后进行播放。此时多媒体文件的剩余部分将在后台的服务器内继续下载。

常用的流媒体协议有实时协议(Real Time Protocol，RTP)和实时流协议(Real Time Streaming Protocol，RTSP)，参见本书第 7 章的详细描述。

流媒体的应用主要有两类，一是已有媒体内容在需要时使用，如视频点播；二是将内容直播给大量用户，如 Web 直播。这两种使用方式对网络、硬件和软件的要求完全不同。在第一种点播的情况下，每个用户和服务器之间建立连接(如图 5.14 所示)，这样导致服务器的负载和带宽成为主要问题，而广播对网络层的需求更高。理想情况下，一个用于广播的服务器管理一个媒体流，由路由器等网络底层基础支持，将媒体同步播送到所有用户(如图 5.15 所示)。而实际上，因为 Internet 一般并不支持多播技术，所以和点播类似，服务器必须使用 P2P 连接，并模拟广播功能。

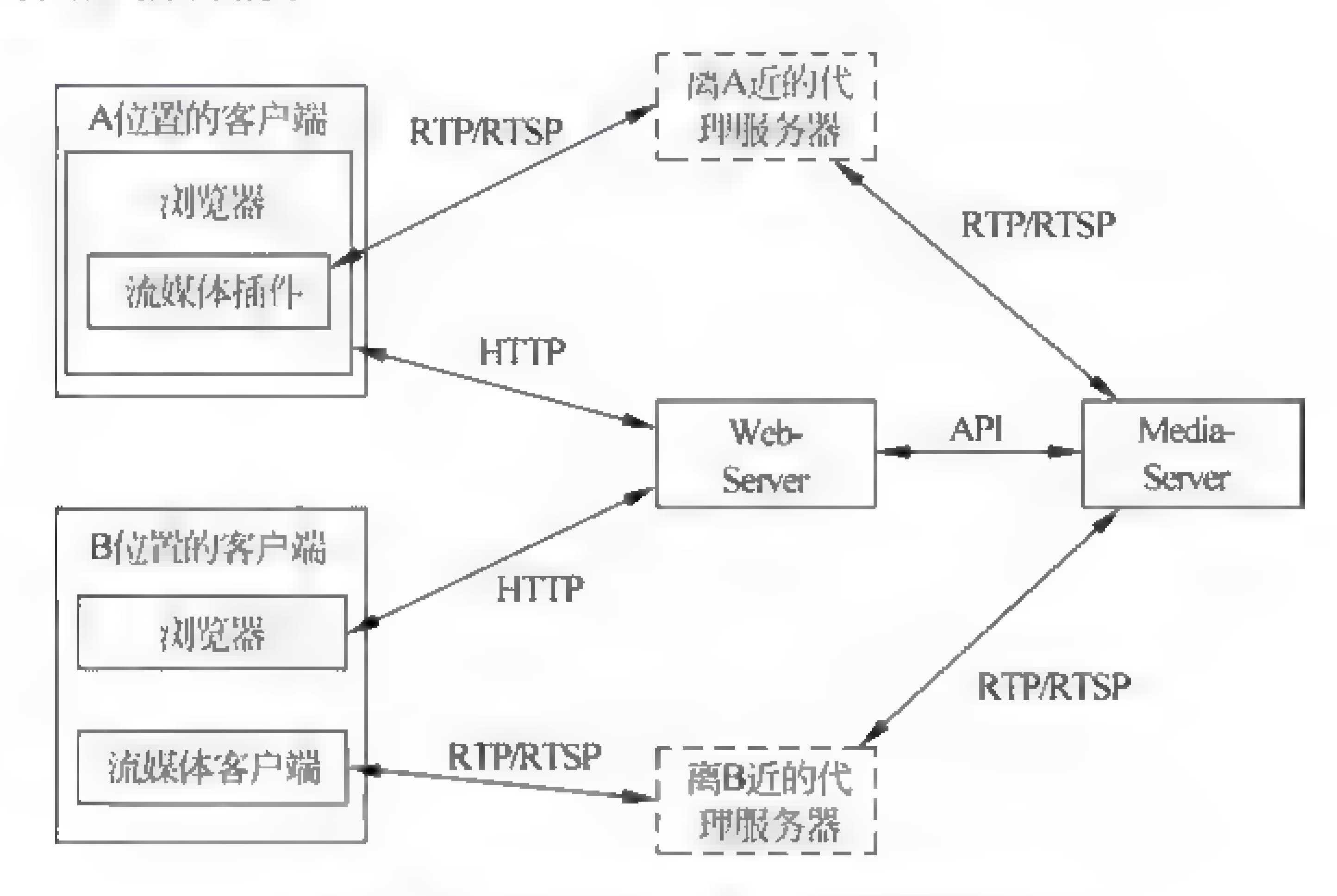


图 5.14 使用点对点连接的流媒体架构

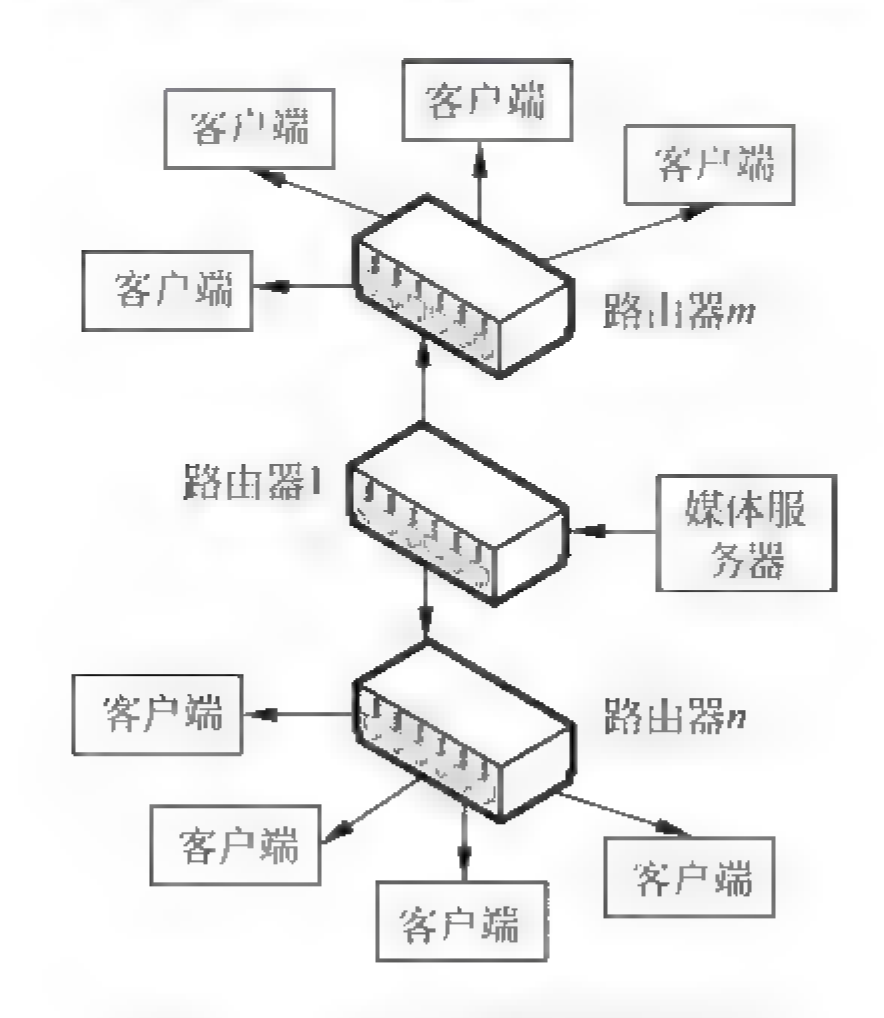


图 5.15 使用广播的流媒体架构

为了使这类系统能支持大量用户,有必要将多媒体数据分布缓存在缓存服务器,给用户提供尽可能短的访问路径。这种方法将用户的请求转发到离其最近的包含所请求数据的结点上。这种内容分布本身既可以是推送方式也可以是主动抓取方式,如在缺乏地理位置信息的情况下,只有当用户请求数据时,才会将内容发送到相应的结点。

流媒体架构目前还面临一些挑战,如对于预推送分布的策略的制定以及相应的分布式基础设施的代价相当高,而且维护量也大,因此,目前对于推送方式的使用并不普及。媒体的交互性差,如浏览器的插件(如苹果公司的 QuickTime Player 或微软的 Windows Media Player)一般只是用于播放多媒体内容,而无法刷新连接。对于时间敏感的媒体,HTML 页面的文本和图片等内容不能和媒体内容同步。针对这些问题,出现了同步多媒体集成语言(Synchronized Multimedia Integration Language,SMIL),以改善交互方面的局限。

5.6 总结与展望

Web 应用日益复杂,需求复杂多变,单纯依靠某种先进技术已经无法达到快速开发、验证和部署。良好的 Web 应用架构是 Web 应用存在的必要条件。好的架构可以提高 Web 应用的开发效率,提高 Web 应用的可重用性,易于维护和扩展。因此,将 Web 应用开发技术综合起来,根据特定应用将系统逻辑进行层次化组织,或者集成,形成完整的 Web 应用架构已经成为 Web 架构发展的趋势。

本章重点分析了现有 Web 应用架构,内容涉及模型驱动架构、架构模式、层次架构、面向数据的架构和集成架构等内容。相应的各个方面还会随着技术的发展进一步发展。

客户端设备和服务器端的基础设施的发展,推动了 Web 应用的发展。普适计算和面向门户的应用无疑会不断发展。这种趋势还推动着其他基础设施的发展,如 P2P 对普适 Web 应用的发展有着重要的影响,如以此为基础的 Amason 和人人网的 NoSQL 架构;网格技术构建透明的计算机网络来增大 Web 应用的计算能力(如 Globus 工具包)。

在这些趋势下,Web 应用越来越多地采用集成架构,将已有功能快速集成到新的 Web 应用的同时,更要考虑应用的上下文。而且,集成会逐渐使得客户端和服务器端之间的界线越来越模糊,比如现在开发的 Web 应用中的功能,不久将会成为更大 Web 应用的一部分,典型的例子如搜索引擎提供的服务功能,在 Web 应用中广泛使用。而相应的开发架构的方法和工具目前对 Web 应用的支持还远远不够。

多媒体基础设施的发展速度受到电信业投资高的影响,而数字电视和在线游戏倒是这个领域发展的希望。

第6章 Web应用设计

正如 Jakob Nielsen 在其关于 Web 设计的著作中所述："本质上设计有两种基本途径：表达自己的艺术设想和为客户解决问题的工程设想。"设计从艺术想象中发展而来，而艺术想象本身又随着 Web 应用的构建而发展。在 Web 发展的早期，支持超文本系统简单的链接概念的基础上，很多开发者选择艺术设想，以一种特别的方式进行设计，而且经常在生成 HTML 时才进行设计。

随着 Web 应用的成长和成熟，虽然 XML 等技术逐渐被使用，但还不够广泛，Web 应用相关的设计大多还停留在基于数据库的设计，即信息设计。在 Web 应用设计过程中，除了信息设计之外，还需在客户端和服务器端扩展集成软件模块，因此面向对象软件设计也在 Web 应用设计中起着举足轻重的作用。不论哪一种方法，都无法满足 Web 应用的特性，而 Web 应用还不够成熟，专门针对性的最佳实践、好的设计方法、设计过程、设计符号和设计工具等还没有发展起来。

Web 应用设计是在 Web 应用分析模型的结果之上，形成 Web 应用的设计方案。除了包括功能设计和信息设计之外，还包括用户和页面的交互设计、页面（即展示）设计。所有的设计的目标是提供好的用户体验。

本章推荐将 Web 应用分为不同逻辑层进行设计，而且又考虑两个设计部分：组件和网。组件主要针对 Web 应用结点，即媒体、软件组件、数据库访问等，而网用于这些组件的布置。还考虑到 Web 应用设计受实现技术的限制，所以本章中会多次出现对第 7 章 Web 应用开发与部署的关联。



6.1 Web 应用设计特性

万维网问世之初，主要是以文本加上链接的方式（即超文本）组织和展示信息。用户可以通过 URL 访问这些信息。这类简单 Web 应用的设计主要是信息设计，即创作者设计内容，程序员采用 HTML 进行技术实现。HTML 在标记的基础上，逐步增加图片、视频和音频等的支持。超文本文档由以下两种内容组成。

（1）结点、链接和锚。

（2）网和其他聚合体。Web 早期，网由结点和链接关联组成，称为超文本文档，如预置阅读路径可以像结点一样嵌入其他更大网的元结点。

由于 HTML 初期只关心创作特性，具有非线性特点，其简单且免费使其在早期的 Web

中广泛使用。但就设计而言，其主要缺点如下。

(1) HTML 只是在文本文档中加入一些标记，导致人们忽视结点的原子性，文档(结点)长度难以控制，非线性阅读特性不够成熟。

(2) HTML 通过链接和锚标记等确定超文本结构和文档的标题以及列表等，结构、背景颜色和字体等布局混杂。

(3) 如第 5 章所述，尽管 Web 已经区分分布式软件架构，将浏览器和服务器区分开来，但是，HTML 缺乏软件架构的抽象机制。

(4) HTML 是以文本为中心，很多其他媒体无法由链接来支持，只能通过链接目标来实现。

(5) 对结点和链接类型的自定义、反向链接、分离链接和结点的存储、不同锚等仍然无法由 HTML 支持。

随着互联网的发展，这种简单共享信息方式无法满足用户需求，因而，基于 XML 的标准得到不断发展。相比 HTML 而言，XML 除了定义有效标记之外，还定义可用于整个文档类的规则，即文档类型定义(DTD)，后来发展出 XML 模式(XML Schema)。相比而言，XML 的最重要的特点是其文档类型和可移植性。SOAP、XHTML 等简单程序设计语言也基于 XML 而生。HTML 的缺点在 XML 时代得到解决。

从创作内容的角度而言，设计基于文档的 Web 应用应该遵循如下一些基本规则。

(1) 网应该是信息设计的核心。

(2) 传统文档应该分解为原子结点。

(3) 即使技术上不支持布局与内容、结点和网等的分离，也应该在概念上进行区分。

(4) 所选技术应该支持如集中链接管理、内容管理等高级概念，至少是在设计时支持，应该优先考虑基于 XML 的解决方法。

XML 语法可以从形式上定义，而非语言上，浏览器可以解析 XML 模式和文档，但只能执行 XHTML。因而，虽然基于 XML 的标准逐步发展，但发展速度受到浏览器和 Web 服务器的限制。在浏览器端执行的脚本语言 JavaScript、服务器端接口 CGI 等技术，提供动态创建 HTML 文档能力，成为"动态"的第一步。Java 技术的出现，使 Web 应用的软件特性得到进一步加强，具有了可编程 Web 特性，浏览器中不仅可展现 HTML 文档，还可以执行 Java 程序(Java Applet)。Internet 上分布式程序的发展，IPC、RPC、CORBA、RMI 以及 Web 服务等技术在各种分布式系统中不断使用，使得 Web 应用设计体现出更多的软件设计特性。

不管 Web 技术有没有和分布式程序设计技术完全合并使用，都可根据 Web 应用的各种特性，将 Web 应用设计分为子任务进行描述。在描述时，又区分 Web 应用的组件(即结点和链接)和网(包含组件的整个 Web 应用)。

敏捷开发方法的极端推崇主义者会认为 Web 应用应该只是进行有限的设计，而这对于特别简单的 Web 应用来说适用。然而，目前的 Web 应用日益复杂，具有很大程度的软件特性，因此，即使是轻量级的，也需要对 Web 应用进行设计，也可以和原型一起进行。当然，Web 应用的需求易变性特性，需要考虑设计的详细程度。

在第 4 章中强调了面向数据将建模分为内容、超文本和展示三层。Web 应用设计关注 Web 应用更多不同方面。因为美学、内容和技术适当结合会因 Web 应用特性和复杂度的

不同而异,所以设计对每个项目都有区别。可以根据Web应用的普遍特性,将Web应用分层进行设计。针对Web应用复杂性的快速增长,本章将Web应用设计的最底层定位为侧重软件特性的功能设计。即将Web应用设计分为交互设计、展示设计、内容设计和功能设计。这一分层方式有助于模块化和功能化设计,并且可用于所有层。每层设计又考虑组件和网两个设计部分,也因此,Web工程的一个重要技巧是对这些不同元素的有效集成。

Web应用设计的最终用途是为了满足用户的需要。随着Internet应用的普及,计算机的主要作用已逐步从以计算为核心向以通信为核心转变,用户上网的目的已不再单单是从事计算功能,更多地是为了网上浏览、发送文件、E mail、下载、看电影等。计算机已经变成了人们生活空间的一种扩展,Internet和Web为人们提供的不再是简单的计算网络,而是一个信息源,一个丰富多彩、无限扩展的生活空间。在人机交互的空间里,机器只是一种工具而已,人才是核心。

最终用户每天都会使用Web应用,因此,从对最终用户来说很重要的问题开始设计。Web应用建模时曾经提过,Web应用至少一个参与者是“人”,因此,几乎所有的Web应用都有一个交互性很强的组件。交互设计是用户接受的枢纽,也是Web应用本身成功的关键。因此,Web应用设计从交互设计开始。交互设计主要关注用户和Web应用交互的控制流。

Web应用能否在一瞬间抓住用户眼球,吸引用户继续访问,取决于页面。用户总是喜欢有好的观感,即布局合理、外观漂亮、使用方便、信息丰富的Web应用。Web页面设计决定了Web应用的形式。展示设计关注组件部分的文档、媒体、数据等的输出展现方式,网部分关注当前访问用户的视觉、听觉、多种形式的输出。

上述将展示和交互明确区分开进行设计,类似于MVC模式的概念及其视图和控制器的扩展。一方面展示网与组件的关系设计,强调内容有关的多媒体特性,以及艺术性和自我解释性有关的展示特性。因此,设计时媒体设计人员、Web应用开发人员等需要相互协作;另一方面,在网中交互,即导航,可以通过浏览来表达。用户完全随机地使用链接进行导航,部分也可以通过软件功能进行调整,不同场景提供不同的导航选择,如动态变化的导航条和图形布局。

Web应用如果仅有漂亮的用户界面是无法长久地吸引用户访问的,除了引人注目的Web页面外,Web应用还必须要有好的内容并以清晰、高效、便捷的方式为用户呈现这些内容。条理清晰的Web应用内容架构便于用户从庞杂浩瀚的信息中找到自己所需的内容,再加上易于使用的内容导航等,只有将这些都结合起来,才能留住用户,增强用户的忠诚度。

Web应用复杂性的提高,已经逐渐发展为除了提供各类内容之外,还针对各种应用领域提供大量不同的功能。例如,电子商务应用除了提供如产品描述、规格说明和照片等大量的内容外,还提供如用户定制、订购产品等很多功能。某些情况下,功能将采用既定模式。例如,B2C(Business-to-Consumer)应用使用目录、购物车和付款台。另外一些情况下,功能可能相当独特。功能设计和内容设计类似MVC中的模型部分。虽然面向对象方法可以将功能和数据不同方面集成在一起进行设计,但是我们在6.5节Web应用的功能设计中的描述并不受限于某种开发策略,而受到信息系统和数据库以及Web应用分类的影响。

技术的发展,也影响着Web应用的设计。Java和ASP/.NET等对Web应用的功能实现与扩展起了很大作用。W3C标准加强了Web的交互性。主动组件的重要性体现在如

SOAP标准实现的远程过程调用，以及如UDDI的一个简单"静态"链接，实际上其链接目标在运行时才能确定。受信息系统的影响，网的一端具有工作流管理的特性，可以将业务过程分步骤进行设计，每个步骤又可以设计为组件。加上Web分维概念的引入，业务过程之间可以跨组织协作，如W3C 2002a所述业务流程。随着服务开放，增大了普适Web应用的动态程度，Web组件看做服务，可灵活方便地根据用户需求集成到普适、可定制的复杂Web应用中。

最后，需要区分出设计的结束，简单来说就是终止于构建。但是需要注意，Web应用的开发过程中设计和构建之间经常会交织进行，也就是说，设计会延续到构建，而构建会关注交互、页面、内容和功能将如何实现。

6.2 交互设计

交互设计涉及Web应用的可视性、动态性、功能性和技术性元素的交织，其目的是很好地将这些元素综合并解决相互之间的冲突，以提供给用户感兴趣的、有吸引力的、一致的和易于理解的体验。交互设计由人机工程学、人机交互界面、可用性工程、视觉传达设计、认知心理学、工业设计、用户体验设计等多门学科交叉组成。本节将Web应用的交互设计系统地划分为4个方面：用户交互、用户页面组织、导航和用户活动。

6.2.1 用户交互

从Web作为信息传播的工具这个角度来看，可以把交互定义为信息的互换、互动和反馈，即需要用户做出某种动作，如填写表单、点击链接或按钮等以实现某种目标或进入下一步。交互设计是人工制品、环境和系统的行为，以及传达这种行为的外形元素的设计和定义，它首先旨在规划和描述事物的行为方式，然后描述传达这种行为的最有效形式。

从交互的角度来讲，可以将Internet理解为学习者和学习者的知识之间的抽象界面。Web应用交互设计是人与计算机交互方式的演变，是随计算机技术的发展而发展的。从计算机诞生到现在，人机交互的方式从利用穿孔纸带输入计算机程序，发展到面对终端机上的字符操作界面，再到个人计算机上图形界面的多媒体。

许多具有Web-enabling特性的遗留系统或应用将交互设计合并到展示设计中，这意味着这些系统知识简单地将其展示方式"翻译"成HTML页面，再引入很少的如并发数据库访问等其他功能。

随着Web应用变得越来越复杂，HTML担当的角色越来越多：信息传递、布局、用户交互、导航、过程（链接变化序列）和对数据内容的直接访问。随着HTML职责的增加，它日益发展，包括了更多的功能以应对日益复杂的使用场景。在这一过程中，用户交互成为最主要的障碍：服务器每次都需要产生新页面，应用运行速度慢，而且由于表达不足以提供更高级的用户交互技术，HTML很快就体现出其与桌面应用之间的差距。

为了克服HTML的这些障碍，出现了很多相关技术，从发展的历程来看，交互方式越来越多地考虑到人的因素，日益朝着更友好、更便捷的方式发展，但还没有出现哪一种方式体现出了其适应开发需求的优势。但是，到底需要多少页面功能来显示数据和执行操作，

Web应用的数据到底有多么密集，这两者之间要达到平衡，就要看其移植性和技术是否满足提供商以及用户和(或)客户的满意。因此人性化的设计是Web应用设计的核心，如何根据人的心理、生理特点，运用技术手段，创造简单、友好的交互方式，是Web应用交互设计的重点。Web应用设计的标准却可以回归到软件设计的主要属性：可维护性、可重用性、可伸缩性、可扩充性和可持续性等。

可维护性指当应用程序失败时定位和修复的容易程度，通常用简单性、一致性、模块化和自我描述性来体现。对于Web应用而言，采用ActiveX Applet或AJAX等技术，使其页面具有很大吸引力和很高的用户交互性，这样的页面上展示、数据和逻辑通常是紧耦合的，其开发和维护难度很高。DHTML和Portlet等技术将这几个方面分开考虑，其模块化和可维护性相对高一些。

可重用性指一个应用中的代码在另一个应用中无须修改就能使用的可能性。对于Web应用而言，一些开发技术提供了不同的重用机制，如代码库或者脚本库。但是由于Web应用开发时需要快速创建出页面，所以经常会忽略掉是否有页面能够重用。

可伸缩性不仅要支持大量的用户，而且也需要能够识别哪些不同开发活动可由开发团队并行进行开发。如在一个大型项目中，一些基于ActiveX技术的功能因其耦合性使得难以同时进行开发，而如XHTML和XSL/T等基于XML的技术从开发的角度来说具有更好的伸缩性。

另外，在用户交互设计时也需要考虑可扩充性，即更好地改进应用的功能的效率，以及可持续性，将在6.6节中进行描述。

6.2.2 用户页面组织

由于用户页面本身是展示给用户的，所以它和展示设计联系非常紧密，不过，我们在这一部分主要关注集成方面，而非展示方面。Web应用的应用页面经常需要展示大量的信息、信息上的操作，以及信息之间的关系。如何将所有这些都联系在一起进行设计具有很大的挑战。解决这一问题的首要步骤是将这些元素进行分组组织，而且分组必须在整个页面中清晰而一致，大部分输入和交互组在一次会话中应该是一致的。

当结点包含很多信息而无法适合一屏时，需要在展示设计和交互设计之间进行权衡。最好衡量屏幕大小和结点原子性哪个具有更高优先级，结点是否可以拆分成几个更小的结点，是否可用更多的导航来代替滚动，用户页面的复杂行为和可移植性之间是否能达到平衡。根据导航语义(即导航是触发继续阅读还是访问相关主题等)、可移植性以及可用性，可以有如下不同的方法。

(1) 结点一次性以HTML全部发送给用户，HTML页面既包含脚本又包含插件技术来使用户访问信息的一部分，这种嵌入程序设计避免了更多的导航。这种方式具有较好的导航语义和可重用性，但是可移植性就差一些。

(2) 结点一次以大HTML页面方式发送给客户，不包含脚本，用户选择导航链接在页面内导航。这种方式的导航语义和可移植性较好，而可重用性差一些。

(3) 结点一部分发送给用户，页面显示有意义的部分信息，其余的用户感兴趣的信息页面由用户自己导航获取。这种方式可移植性好，而导航语义和重用性差。

页面加链接的方式避免了滚屏，但是增加了导航以及更多延时。上述各种特性需要根

据Web应用的特性总体进行衡量，如对于企业内部使用内联网的应用可能假设使用同样的浏览器，而电子商务提供商则必须考虑可移植性使所有潜在客户都能够访问其页面，以做出更加合理的设计。

6.2.3 导航设计

导航设计主要完成用户可以访问的元素和导航结构的设计。最简单的情况下，元素变成结点，结构定义结点之间的关系。这些关系最终会在用户页面中体现成链接锚。这种场景下，交互设计定义导航自己所需要的部分(锚和URL)以及用户引导他们自己的元素。

设计链接的代表——锚，是URL的可视化展示方式，因此，我们必须理解用户的行为和可能的后果。因为基于HTML的Web实现和锚概念相混合，成为无方向性链接元素<a>，其语义也无法再区分。这也就是为什么用户无法肯定跟着链接走的后果是什么。当<a>表示导航时，链接是表示一个或多个目的地，是在站内还是站外，页内还是页外，会打开新的浏览器窗口还是在同一个窗口中，等等；当<a>表示下载时，下载的文档类型是什么，是否有表示文档需要的工具(如插件)；当<a>表示处理时，链接是否会触发服务器端动作，能否导航回来或取消动作。

锚的文本最好是可以自我解释的，可以在锚内采用图标来使链接可用。这类锚和图标可以静态设定，通过在页面中嵌入脚本来动态设定属性，如某些媒体类型是否可以打开等。

导航过程通过锚在用户界面中触发。这些锚表示成链接(HTML页面中的URL)，指出导航过程的目标，因此需要清晰、一致、长期存在以及指向绝对路径。

XML的出现意味着链接和锚的一大进步，结合XPath、XPointer和XLink等XML标准，可以提供通用超媒体基础，远比HTML更进一步。Web应用设计者应该了解XML中链接的扩展能力和自我解释能力等，如多方向链接(Multidirectional Links)支持一个XML文档可以链接多个资源；XLink既支持外部链接，又支持内部链接，还可以为文档定义相关链接数据库。

导航设计还需要关注导航及其方向。导航工具应该有助于减轻用户的认知压力。为了达到这个目的，可以从如下三种基本策略着手。

(1) 导航组织策略，用于确定整体导航结构。

(2) 方向援助策略，解决展示设计的交互方面的"我在哪儿?"和"我访问过哪儿?"等问题。

(3) 链接认知策略，主要关注链接关联有关的目的和结果。

这里主要考虑第一种策略：导航控件组织，支持有意义导航活动，如避免导航冗余。以搜索结果为代表的索引，也称为星型(Star-Shaped)导航，用户必须从某个索引项返回后才可以再选择其他索引项。

随着Web应用规模逐渐增大，用户获得越来越多的方向和导航帮助。例如，一个总是活动且可认知的对象，用于进一步导航其他对象(结点和子索引)。这个活动引用对象，包括目标对象的展示，保持可见，有助于用户查看目标对象或者选择相关对象。关于逆向导航，不仅需要指向当前位置的路径信息，还需在路径中包含通向索引和结点的缩写。

6.2.4 复杂活动的会话设计

在设计复杂Web应用时，经常需要用户参与才能触发一些扩展步骤的可视化。扩展可

表示一个活动扩展到多个页面。这种情况下，可以将前向导航分为三种：①导航步骤触发一个动作；②导航只调用一个页面，如一个表单的第二页；③导航步骤导向一个并不直接参与活动的结点，如其他信息等。这些不同种类的导航既缺乏一致性又迷惑用户。比如用户和Web应用之间达成协议的checkout（结账离开）操作，如果用户决定到另一个单独的页面完成checkout，那么必须能够明确当前的活动状态，当用户在没有checkout的时候单击"后退"按钮时，也一样需要区分。

这些问题说明，从交互的角度而言，业务过程和超文本应用的特性相差甚远。从控制、离开页面/活动以及撤销/继续三个方面列举如下。

(1) 超文本。用户控制访问页面序列，页面通过链接到达；用户通过选择锚离开页面，而页面的状态并不改变，不能结束页面；通过如"后退"的方式返回一个页面仅仅意味着页面再次被调用。

(2) 业务过程。定义序列的下一个活动，活动按序执行，但是控制流可以很复杂；当离开一个活动，需要明确是已经结束还是中断或终止，活动结束是业务过程非常重要的一部分；返回到一个活动意味着中断的活动状态要继续，要终止一个活动，需要明确调用。

Web应用中实现业务过程既可以通过简单HTML完成，也可以是工作流管理的方法和工具。过程定义通常描述过程是如何嵌入工作流中，而不是用户如何和业务过程交互。因此，交互设计要尽量将复杂任务和导航的可能性关联起来。

6.2.5　交互设计原则

不管是重业务过程还是重超文本的Web应用，在进行交互设计时，首先需要考虑与技术和架构的相互作用和相互影响。Web应用开发过程中，设计、架构和技术之间关系紧密，尤其是从简单到复杂活动的转换影响着对架构和技术的选择。有时随着Web应用的发展演化，会转换到复杂架构和采用更好性能的技术，这种转换也会对Web应用设计产生影响。

简单地获取信息的活动可以通过3层架构来实现，采用模板生成客户请求的HTML输出，如基于ASP.NET、JSP、PHP或Ruby等。采用这种简单架构，应用控制和应用逻辑通过脚本代码嵌入到模板中。

而随着信息的表达变得复杂时，如从多个信息源中获取信息，脚本也会变得异常庞大，此时，最好通过采用用户定义的服务器端标记替代脚本语言。这些标记使用户能够从HTML页面中分离和隐藏代码。不过，即使将HTML和代码分开来，用户定义标记中的控制逻辑仍然在每个结点中分开实现，每个标记单独确定后续显示序列，并导向合适的模板。这一概念类似于C语言中的"goto"命令，其缺点是维护性差，难以理解，模块性差。因此，对于交互复杂的应用，尽量使用类似MVC的架构。

对于复杂业务过程的系统而言，MVC却又存在不足，它不支持需要多个用户输入的事务，因此这类事务无法扩展到多个结点（如页面、动作），限制了应用逻辑中包含复杂事务。改进控制器可以用于支持业务过程，Java EE架构中将控制权分为前端控制器(FrontController)和视图分发器(ViewDispatcher)，.NET框架在ASP.NET中利用继承概念将所有控制器的公共行为封装为一个联合页面控制器(PageController)，所有页面都直接或间接继承自这个页面。

交互设计的好与坏，最有发言权的是用户。因此，尽快将交互设计方案转化为可以让用

户交互的原型，让用户亲身去感受，切实地对Web应用进行操作与运行，结合他们的期望与实际Web应用的用户体验，进而尽可能地开发出满足用户需求的Web应用交互方式。一般情况下，交互设计需要遵循如下几个方面的指导原则。

1）控件控制

（1）第三方服务器端控件的使用要保证具有广泛兼容性和安全性，且具有完备的接口指定外观属性和交互方式。

（2）复杂的应用程序中，非标准交互控件给出详细的操作方法的提示。

（3）页面中尽量使用一致的导航类型，如使用基于点击"图形"的链接、"文字"的链接或文字图形混合方式。

（4）页面按钮作为基本交互控件，提倡使用有鼠标响应状态变化和禁用状态的BUTTON按钮，除特殊界面需要，不提倡使用图形按钮，而且保证同一应用程序内同类功能按钮只使用一种外观。

（5）服务器端控件在交互过程中不能破坏页面表格原布局，如.NET中CALENDAR控件最好在弹出的子窗体内独立使用。

（6）包含数据的表格中，没有数据的情况要有文字标注（如无**数据），表头字段名用区别于数据行的格式显示。

（7）分栏目的主题名称使用用户容易理解的名称，以用户第一人称角度的命名方式，减少生硬的称谓给用户带来的不友好感。

（8）Web应用允许含有类似拖放操作的非标准交互控件，但是需增加操作的说明。

2）表单控制

（1）页面内部有必要的前后文帮助信息，将页面主要任务目标、注意事项等描述在表单前申明，便于用户及时获得导引；上下文帮助文本中保持业务术语前后一致，避免使用计算机专业术语。前因后果一一呼应，保证用户在最短时间内解决交互中的疑问。

（2）页面在交互控制中添加完整的状态控制，操作中灰显特定组合的控件来实现用户的准确操作，及时地刷新状态变更后表单内的数据及状态。

（3）表单内任务无关的信息、较少使用的选项等可以通过DHTML技术、服务器端控件的隐藏等减少用户操作中的干扰因素。

（4）表单内，在特定的字段域附近给出必填信息提示，并用醒目颜色标注，提醒用户注意，验证的错误提示要给出准确恰当的指导；为提高用户填写的效率，建议使用客户端验证；复杂逻辑的验证使用服务器端验证。

（5）信息显示过滤可能出现的用户不能识别的HTML特殊字符。

（6）表单中用户在交互过程中保证用户方便地切换编辑、浏览状态，方便用户用最快的速度获取需要的信息，提高操作效率。

（7）经常使用的工具按钮（如新增、编辑等功能按钮）保证在页面经单向拖曳浏览后，无需来回拖曳，滑竿即可操作；长页面可以考虑页首、页尾均放置工具按钮。

（8）主详细表及父子表关系的查看方式使用联动式导航到下级数据，即点选主项目或父项目记录时系统自动查询并显示出关联的详细信息、子表数据，无须单击任何按钮。

3）窗体控制

（1）使用具有广泛兼容性的JavaScript控制客户端交互和简单导航，除服务器控件部

分自动扩展到客户端的 JavaScript 外，Web 开发人员手动控制脚本不推荐使用 JavaScript 和 VBScript。

(2) 操作过程中有清晰分界的子任务使用弹出窗体或分页框实现，保证各显示窗体的数据同步，及时刷新主任务窗体，使用户看到操作完成的结果，并且通过控件获取焦点等措施突出显示该结果。

(3) 窗体(或 Frame)内页面发生切换时努力使用户需要记忆的中间结果为最少，甚至没有。

(4) 弹出窗体的页面主题、栏目标题(Title)等资料与关联的父窗体保持上下文一致，方便用户理解并做出处理策略。

(5) 采用框架结构的应用程序，要充分考虑不同分辨率下的自动扩展。不同框架之间同步通信及时，方便用户快速切换目标导航，观察数据之间的关系，等等。

(6) 窗口主题显示标志用户当前所在模块或子系统名称，子任务窗体主题使用“动词＋名词”的语法结构指明用户当前的任务。

4) 输入设计

(1) 尽量使用能够自动纠正或提供正确导引的输入方式，提高效率和避免发生错误。如录入日期使用用户单击弹出的日历控件自动返回正确的格式，无须干预。

(2) 方便地获取到必要的信息，无须用户记忆中间结果。

(3) 表单格式尽量保持业务原始票据的格式或字段排列顺序，方便用户的集中录入过程。

(4) 表单字段左对齐(根据不同语言习惯而定)。

(5) 输入控件的宽度基本符合数据库能够容纳的宽度，暗示系统能够接受的字符容量。

5) 提示信息

(1) 错误操作的提示信息使用非专业的、易理解的名词告知用户。

(2) 以第二人称“你”或“您”称呼用户。强调用户的主导能力。

(3) 对用户宽容的语气。

(4) 严重的警告信息使用弹出信息框提示，不严重的在页面前后文处直接输出，弹出不宜太频繁地使用。

(5) 可能对系统导致破坏性的操作要给出警告信息和用户“确认(Confirm)”按钮，用户可以“取消”操作，防止意外错误操作造成损失。

(6) 复杂步骤在完成后给出“完成”提示。

6) 出错处理及出错页面的跳转

(1) 在系统的内部状态变化对用户有较大影响的情况下，给用户明显的解决提示，或给出自动的导航，使用户快速地恢复工作状态。

(2) 用户状态过期，用户无法进行操作时，系统自动跳转至登录界面。

(3) 提供应用程序级的错误截获，在不可预见的情况下给用户告知当前情况。

(4) 提供页面间自动导航控制，以更宽容的方式接受用户操作，协助用户处理复杂的交互任务。

6.3 展示设计

展示设计指媒体设计者定义视觉和某种意义上多媒体内容的表现形式，是交互设计的延伸，是一项重要的活动，主要关注用户界面的结构和组织形式，包括屏幕布局的展示、交互模式的定义和导航机制的描述。除了Web应用的功能、对工作流顺序和相关问题的描述之外，还有美学设计(也称为美术设计)，描述Web应用的外观和感觉，包括颜色配置、几何布局、文字大小、字体和位置、图形的使用和相关的美学决策。给Web应用设计合理而友好的页面就像人的衣服一样，合体舒适的搭配能给人耳目一新的感觉，留下好的"第一印象"，并在第一时间吸引用户。如果Web应用页面设计不好，那么，无论Web应用的内容价值、处理能力、服务的复杂性以及自身的总体收益如何，由于在实际的每个领域都有众多有竞争力的Web应用，所以很容易令人敬而远之，甚至丧失信心而转到其他Web应用。

设计的一切活动和成果都为人服务。如何根据人的特点，运用技术手段，创建简单、友好且令人满意的人性化的Web页面，是Web应用展示设计的重点。

从重内容的角度而言，展示设计最初采用HTML，混合了内容、格式、链接甚至脚本程序。而现代的展示设计概念是将Web应用内容和展示分别进行设计。Web应用的多媒体内容从组件角度进行开发，并隐含地在网中进行定义，这就意味着好的展示设计应该针对各种文化、技术和上下文需求，具有很好的适应性。在Web应用生命周期内，为了提供新的视觉设计，Web页面甚至是整个Web应用需要改版或重新构建。如果采用传统Web开发方法完成这项工作，则需要对几百或几千个HTML文档进行修改，即使有一些工具做支持，也只能完成部分工作，其余部分仍需要手工完成，这需要用户具有扎实的HTML知识，对于大型系统而言代价相当高。

对Web应用进行展示设计的工具很多，如美国Macromedia公司开发的Web页面制作和Web应用管理的Web页面编辑器Dreamweaver，微软的FrontPage，可视化的Web应用的构建和管理工具NetObject Fusion，Adobe系列，CorelDRAW图形套件，等等，大致可以分为两类：页面编辑器和内容管理系统。页面编辑器主要用于创建小型Internet展示，主要优点是类似于软件标准，使用户能够在熟悉的环境工作；主要缺点是需要HTML知识，关注页面层设计而无整体概念，布局、导航和交互混为一体。这只适合于小型Web应用。内容管理系统将编辑活动和布局分开，易于维护Internet展示，缺点是需要将这些展示进行映射，优点是只需要为各种参与角色(图形艺术家、编辑等)提供工具，而一般不需要HTML知识，并且内容、布局和导航相互分离，一个信息单元的内容可以映射到工作流中。目前，页面编辑器逐渐加入了内容管理系统的功能。

Web页面设计需要考虑业务相关者的需求、用户的认知能力、技术问题与非技术问题、开发人员和用户的经验，以及相似Web应用的经验教训。如Web应用供全球使用，拥有来自不同国家的用户，Web内容和展示可能需要进行本地化，也可能需要在Web应用中使用多种语言。

6.3.1 Web页面特性

通常Web应用由大量的页面组成，每个页面只显示用户当前选择的信息，其他信息根

据导航进行获取。在Web页面设计中，一般存在以下一些问题：Web应用的页面实现的工作量比传统应用界面所占比重加大；Web页面设计语言为浏览器解释执行的HTML语言，界面的实现受到HTML本身的限制；为了使用户感到较好的操作响应性能，客户端界面元素的响应尽可能在客户端实现；Web应用技术发展时间相对于传统的应用程序来说比较短，界面设计还不规范，代码重用率低，一方面Web应用开发人员花费了大量的时间做了琳琅满目、绚丽的界面，但客户使用时却会感到复杂；另一方面每次开发基本上是从头做起，没有很好的重用机制。

与传统应用软件相比，Web应用界面更加灵活多变，运行环境更加多样化，不同之处主要体现在以下几个方面。

(1) 导航更灵活。Web页面中链接是非线性的，这使得用户可以从任何页面跳转到任何其他页面，每一个Web页面都可以作为Web应用的入口；Web页面中导航的表现形式是多种多样的，如按钮、图片链接、文字链接，以及其他事件引发的导航，有些Web页面甚至完全由不同类型的导航链接组成；另外，Web页面比传统软件界面更多地采用了超媒体技术，如音频、视频等。

(2) 多层架构。随着Web应用的不断复杂化，大都采用多层架构。不管是基于浏览器的瘦客户端，还是富客户应用，浏览器端总是负责信息的展示和布局，而应用逻辑、数据以及处理的任务更多在服务器端。Web应用特定的架构要求页面设计应该具有不同的层次。

(3) 页面包容。Web页面有一个界面包含其他界面的情况，用几个现有界面组成新的界面可以提高界面设计的复用性。

(4) 布局不同。传统的应用软件界面通常以任务为中心，围绕一个或多个明确的操作任务来进行布局，界面的元素设置、布局、快捷键设置都是以更高效地完成任务为目标，其界面布局一般是事先进行精确的设置。Web页面在很大程度上是以信息展示为中心的，即使有要求输入的信息，也很少；有关数据处理的任务大多在服务器端，返回的仅是处理结果；Web页面有方便的信息陈列形式，如各种文章名称链接列表，信息文字的长度是不定的，因此，Web页面的布局是即时决定的。

(5) 分工明确。Web应用设计分工明确，每个不同的角色在Web应用设计中都扮演不同的重要角色，如美工设计师负责信息的布局和显示，技术人员负责Web页面的组装和导航，这要求界面描述模型应该明确地分割为不同的层次。

(6) Web页面更易变化。传统应用软件的用户通常可以预先定义，在构建完成之后，用户界面很少再改变，即使有所变动，范围也很小。而Web应用由于其内容的即时性、用户不可知性、全球可访问性等特性，页面变化很多，页面改版频繁，需要有更直观的导航操作指示和简单的操作布局，并考虑界面色彩、动画方面的吸引力。

而相比传统印刷类设计，Web页面设计的新颖性、信息性、娱乐性决定了其多变性的特征，Web页面设计一般遵循如下几个特性。

(1) 新颖性。设计Web页面的目的多种多样，用一种与众不同的方式来表现所要展示的内容往往能使页面传递信息的目的得到更充分的体现。如果页面设计人员设计了一个前人从未见过的页面内容形式，创造了一种独特的页面风格，给了用户一种全新的视觉体验，那么这种页面可能会大受欢迎。

(2) 信息性。Web页面设计可利用Web的无限容量，尽量多放一些别的页面没有的有

价值的信息。信息能做到及时动态更新,准确及时的信息更新是页面的生命所在。

(3) 多维性。多维性源于超链接,由于超链接的出现,页面的组织结构更加丰富,用户可以在各种主题之间自由跳转,从而打破了以前人们接收信息的线性方式。例如,可将页面的组织结构分为序列结构、层次结构、网状结构、复合结构等。但必须考虑页面之间的复杂关系所带来的导航复杂度。

(4) 娱乐性。好的页面生动活泼,富有乐趣。页面设计时往往在内容和形式设计方面都加入了娱乐元素。页面动画为页面设计的娱乐性提供了保障。

(5) 交互性。与传统的传播学理论不同,Web中的传播者和接受者的身份不再明确,信息发布不再是传统媒体供应商的事,任何人都可以参与这个过程,传播和接受信息几乎可以同时完成,人们在瞬间就能进行角色转换。Web内容通过Web页面呈现给用户,每个页面的设计都极具个人特点。因此,在页面设计时必须考虑交互性特点。

(6) 多媒体使用。Web页面中的多媒体元素主要有文字、图像、音频、视频等,并出现了模拟三维的操作界面,在数据压缩技术的改进和流技术的推动下,Internet上已经出现了实时的音频和视频服务。随着网络带宽的增加、芯片处理速度的提高以及跨平台的多媒体文件格式的推广,必将促使设计人员运用多种媒体元素来设计Web页面,以满足和丰富用户对网络信息传输质量提出的更高要求。

(7) 动态性。Web页面会随着Web应用的修改而发生变化,从某种意义上讲,页面设计永远是创作中的作品,随时有用户加以评述、修改和补充。这些评述修改意见可以被附加在文本之后,成为超文本的一个链接,也可以由页面设计者根据用户的意见对文本加以修改,使文本以新的面貌出现。这样页面设计就成为动态艺术。

(8) 虚拟性。页面设计艺术是一种数字化的虚拟艺术,它只存在于网际空间(Cyberspace)中。

(9) 艺术性。Web页面设计是网络技术与设计艺术的交叉,既具有网络的基本特性,又具有平面设计的艺术特性。在页面设计中,技术和艺术紧密结合、相辅相成,将页面艺术设计由平面设计扩展到立体设计,由纯粹的视觉艺术扩展到空间视听艺术,从而更加吸引用户。

(10) 适应性。以用户为中心的Web页面应具有很强的适用性和灵活性,根据用户及其所使用设备的不同特征以及使用活动的不同阶段自适应地给用户提供满足个性化需求的页面。

(11) 可伸缩性。页面发布后,信息和形式的调整要非常方便。此外,页面的交互性使页面信息的发布成为不断更新的循环过程,页面设计人员必须根据用户的反馈信息和Web应用各个阶段的目标,经常对页面进行调整和修改。例如,很多大型Web应用为了保持新鲜感,总是定期或不定期地进行改版,这就需要设计者在保持Web应用视觉效果一致性的基础上,不断创作出新的页面设计。

Web应用设计的诸多特性,使得Web页面设计需要考虑各种组件要素、导航链接网以及其他诸多要素,如用户的行为习惯、页面框架布局(目录结构、页面布局、页面尺寸等)、页面元素(LOGO、图片、动画、文字、点、线、面等)、美学(色彩、风格、创意等)等。只有认真规划,遵循一定的设计思路和策略,才能设计出有吸引力的Web应用页面。

6.3.2 用户行为习惯

"一幅图胜过千言万语"。图能吸引用户，但是由于目前很多应用中，尤其是信息型应用中，有大量广告图片或动画或花哨的字体和格式，大多数用户通过搜索结果点击进入Web应用某个页面，更关注信息而非图像。因此，开发人员应保证Web应用设计凸现出最重要的信息板块。

研究表明，大多用户浏览页面都是首先注意偏左上角的内容，用户的浏览顺序一般是从上到下，从左到右。因此，在构建Web页面时，应该尽量保持这一格式，尊重绝大多数用户的习惯。同时，用户通常匆匆忙忙，大都只浏览页面的小部分内容，要确保标题的代表性，保持信息由简短的段落和句式组成。例如电子商务站点的产品介绍，突出某些部分或者创建项目列表使Web页面信息容易找到和阅读，并保证Web应用内容的要素集中于用户浏览频率高的关键区域，以确保用户的参与。如在新闻系统中，在偏左上的位置放置头条、副题、热点以及重要文章以吸引读者进行阅读。

用户的行为模式直接决定了Web页面的设计方式，因此，设计人员需要遵循这些基本知识，设计出用户满意的Web应用页面。

6.3.3 页面布局设计

对于Web应用而言，吸引用户停留在某个页面上浏览，除了Web应用具有引人入胜的内容外，一个非常重要的因素是合理的页面布局。页面布局将Web页面上的文字、图片、色彩、音频、视频等展示元素合理安排和表现。页面布局并没有统一可接受的格式，本节讨论一些有用的实践。在设计Web页面时，要从整体上把握好页面布局，利用表格或网格把相关Web页面设计的要素协调安排，充分利用并有效分割有限的空间，创建出用户满意的Web页面。

在进行页面布局之前，首先要明确把页面设计成什么样子。线框(Wireframes)描述Web应用内的一个页面(或多个页面)如何从概念上来看达到并行的目的。虽然一个线框并不描述页面的图形设计，但它确实捕获那些应该显示在页面上的核心信息和导航元素以及这些元素的大概布置。本质上，线框是一个没有图形设计或实际实例内容的页面表达，可用于表达内容的集成以及信息结构从Web应用内的一点来看像什么。图6.1所示为幸福密码网动漫心情列表页的线框图设计。

Web应用布局设计时，根据应用特点，选择常见的页面布局类型，如国字型布局、拐角型布局、框架型布局、封面型布局、Flash型布局等。其中前三种布局采用比较固定的布局模式，分别如标题、广告、内容、版权等基本信息，通常页面结构清晰，对称性好，主次分明，缺点是版式呆板，缺少变化。而封面型基本上是用于一些Web应用的首页，类似于杂志封面，大部分为一些精美的平面设计结合一些小的动画，放上几个简单的链接或者仅是一个进入的链接。这种类型大部分出现在企业Web应用和个人主页上。这种布局的优点显而易见，漂亮吸引人，但是缺点是采用了动画等表现形式，使得页面的加载速度比较慢。Flash布局与封面型结构类似，只是这种类型采用了流行的Flash技术，与封面型布局不同的是Flash具有强大的功能。

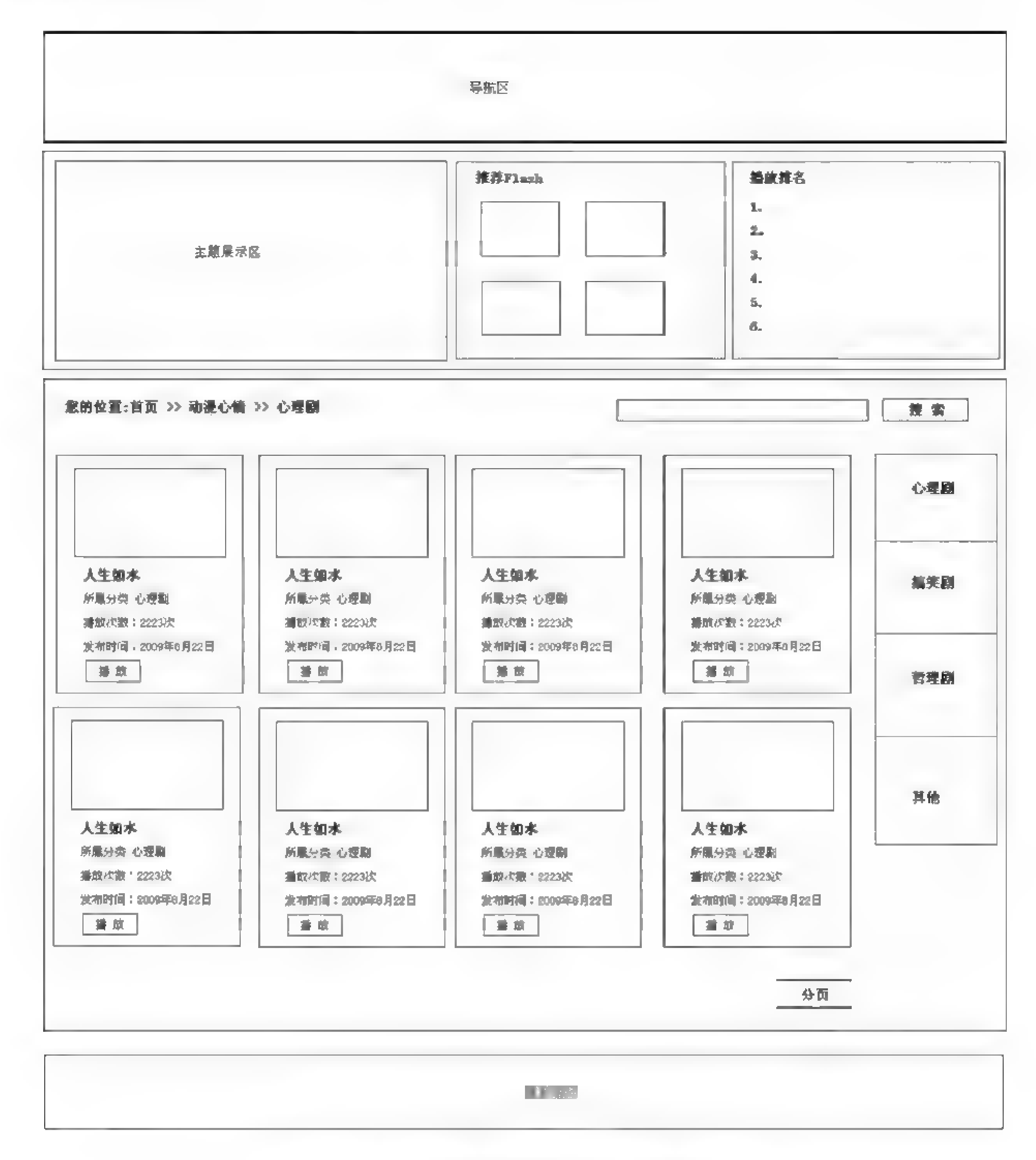


图 6.1 幸福密码网动漫心情列表页线框图

为了实现合理的页面布局，需要采用一些合理的页面布局技术。常见的页面布局的技术有表格布局、框架布局和 DIV+CSS。表格布局是使用最早并且最广泛的布局技术，以<table></table>对数据输出格式进行控制，通过对表格单元格的合并和拆分以及表格中套表格等得到页面布局。框架布局以<frame></frame>为标签，它可以把屏幕进行分割，把不同处理页面放置在不同框架中进行处理，其特点是对于导航或不动的窗口在浏览 Web 应用时只需加载一次。DIV+CSS 与传统的页面布局技术比较起来优势明显，是目前 Web 设计中推荐的方式。CSS 用于控制 Web 页面样式，可以有效地对页面的布局、字体、颜色、背景和其他效果实现更加精确的控制，并允许将样式信息与 Web 页面内容分离。DIV 元素用来为 HTML 文档内大块(Block level)内容提供结构和背景的元素，其中所包含

元素的特性由 DIV 标签的属性来控制，或者是通过使用样式表格化这个块来控制。DIV + CSS 布局具有明显的优势，如大大缩减页面代码，结构清晰，编写容易，表现与内容相分离，轻松控制页面布局，减少了更新页面工作量，等等。

页面布局是决定页面美观与否的一个重要方面，通过合理的、有创意的布局，可以把文字、图像等内容完美地展现在用户面前。在 Web 页面布局的时候，没有绝对的规则，有一些通用的布局原则可以参考，如平衡性、对称性、对比性以及疏密度。同时，还要考虑计算机显示分辨率对浏览器中页面展示空间的影响。

Web 应用的主页(Homepage)的布局，需要进行特殊考虑，它是 Web 应用上查找信息资源的起点，向用户展示该 Web 应用的种类，提供何种信息和服务，等等，为用户指明各种链接路径，提供迅速有效的访问途径。因此，主页的风格样式、色彩布局、栏目设计、文字表述等成为 Web 应用最容易吸引用户但也是最容易产生争议的地方。

主页设计有两种主要的趋势：追求画面美观效果(静态)和追求内容丰富效果(动态)，前者适合内容不多的企业 Web 应用，后者适合内容丰富的综合 Web 应用。不管哪一种趋势，主页设计时在页面版式编排技巧与方法基础上，还必须遵循一些设计的基本原则：机构名称、标识、品牌位置显著，主题风格鲜明，内容明确，易用可读，提供站内搜索。

通过上述页面布局设计，有助于提高 Web 应用的易用性，进而设计出用户满意的 Web 页面布局。

6.3.4 页面元素设计

页面元素是 LOGO、图标、文本块、图片等的组合，页面元素设计即将这些元素遵循设计原则进行组织。而所有视觉设计元素中最基本的元素为点、线、面，将其在二维空间中进行布局。

1) 点、线、面、空间

点是 Web 页面设计中最小的不用考虑形状的因素，可以包罗万象，灵活多变，体现设计者的无限心思，可以是图标、按钮、字母、数字、单词或形状等。在 Web 页面设计中的点，由于大小、形态、位置的不同而给人不同的心理感受。

线是长度远大于宽度的标记，在 Web 页面中使用较多，可用于分割页面。线分为直线、折线、抛物线、自由曲线、弧线以及复合线等。线的总体形状有垂直、水平、倾斜、几何曲线、自由线等。它也是有象征意义的，水平线给人开阔、安宁、平静的感觉；带有锐角的斜线具有动力、不安、速度和现代意识；平行线表现出规律、平稳；垂直线具有庄严、挺拔、力量、向上的感觉；曲线给人柔软流畅的女性特征；自由曲线是最好的情感抒发手段。所以在进行 Web 页面的线条设计时，应该考虑各种线的特点与综合运用。

面的形态除了规则的几何形体外，还有其他一些不规则的形态，表现形式多种多样。面在平面设计中是点的扩大，线的重复形成的。面状给人以整体美感，使空间层次丰富，使单一的空间多元化，表达较含蓄。

二维设计中的空间既指立体感，又指设计师操纵的物理空间。空间的立体感是个强大而动态的方法，使观察者可以深入构图内部。尺寸是建立立体感的最简单的方式，而重叠是营造空间立体感的另一种手段，类似 Adobe Photoshop 等的软件工具可用于营造空间立体感。

Web页面设计中点、线、面并非孤立存在和使用，很多时候都需要将它们结合起来，并存在于空间中，以表达设计意境。使用点、线、面、空间的互相穿插、衬托、补充，构成最佳的页面效果。

2）链接导航

链接导航在Web页面设计中是将无序的网络信息按照一定的逻辑结构或框架组织起来，其本质是一种有效的网络信息组织方式，不仅是信息结构的基础分类，也是用户的路标。当用户进入Web应用，首先会寻找到导航条，了解Web应用储备了哪些分类信息以及分类的方式。

导航分为全局导航、局部导航、上下文导航、辅助导航，有横、竖、多排、图片式、隐藏式等排列样式。由于Web应用可能储备了多种类别的信息与功能，所以一个Web应用不一定只有一个导航菜单，可以有多个导航，会出现两种样式以上的导航风格。在不同风格的页面设计中采用协调统一的导航样式。

作为Web页面的重要视觉元素，应将导航放置在明显、易找、易读的区域，使用户进入Web应用第一时间就可以看到它，并快速了解整个Web应用的内容。导航不是孤立存在的，需根据页面的特性，通过文本或图像来实现，用鼠标触发链接的属性或视觉效果的变化，产生实时动感。导航设计需要注意与页面其他元素协调统一，要遵循一致性、简洁性、清晰性、明确性甚至可追踪性等规范，使用户可以不假思索地快速找到所需内容，进而增强用户体验，提高Web应用的可用性。

3）图片与动画

图片是文字以外最早引入到Web中的多媒体对象。图片的引入使得页面得到了美化，使页面更具吸引力。Web页面中的图片一般采用JPEG、GIF和PNG格式。JPEG格式的图片适用于连续改变显示色调的彩色与黑白图像，比如一些照片、渐变颜色等；GIF格式的图片适合图标、卡通和线条画等颜色要求不是很高、构成比较简单的图片，例如Web应用的LOGO、大的色块构成的图片等；PNG代替GIF格式，比GIF具有可变透明度、跨平台控制图像亮度和可渐进显示图像等优点。不论哪种格式，图片的位置、面积、数量、形式、方向等直接关系到Web页面的视觉传达。在图片选择和优化的同时，应考虑图片在整体页面中的作用，达到和谐整齐，即统一、悦目且突出重点。

动画是传递信息的一种有效手段，可以更直观地解释某个特定内容的操作或者反映某种特定状态，也可以为Web应用添加一些娱乐元素，还可以对Web页面的界面元素使用动态效果，如图标按钮和指针。其基本原理与电影、电视一样，都是利用人类的"视觉暂留"特性。动画通常有GIF和Flash两类。GIF的颜色数少，文件小，网络传输速度影响小。Flash动画具有与多媒体结合，表现力强，体积小，支持脚本语言编程等众多优点，但必须考虑速度的问题。有效的动画设计与其他图形元素一样，需要考虑颜色的协调，声音和动画的流畅等因素。但是，也不要过分地使用动画，否则，可能会分散用户的注意力甚至令用户感到厌烦。此外，应该向使用者提供关闭动画或者其他自定义动作效果的选项。如Web应用使用Flash动画作为入口画面（如shiseido主页），但是入口页面的速度会直接影响用户对该Web应用的兴趣和信心，因而，至少应该提供进度条和略过（Skip）动画的选项。

4）文字

文字是Web应用中信息的主要载体和主要传达手段，是Web页面构成的基础。其表

现的好坏直接影响着用户体验。文字本身也是一种艺术，不同的文字具有俊秀、浑厚、奔放、飘逸等各种风格。文字信息阅读的舒适程度直接关系到用户的心理感受，因此，字体、字号、行间距、段落安排等，都需要精心设计。

字体有各种各样的造型，不同造型都有自己的个性，如宋体的端庄、隶书的古雅、黑体的明朗等，配合不同的主题和页面风格要选择与之相切合的字体。字体的大小和粗细，会给用户文本更重要或更不重要的感觉，并且会使用户察觉出应该按照什么顺序阅读文本。字体的选择不可过多过杂，一般为1～3种，使用太多的字体或样式会造成视觉上的混乱；不要使用变形的字体，变形的字体通常不美观；字体的选择还应该与Web应用内容的性质相吻合，例如，政府页面，其文字具有庄重和严肃性，字体规整简洁。排版时要注意：大标题，小正文；粗标题，细正文；色彩与背景有一定差异，保持可见性、易读性。字大小要适中，较大的字体可用于标题或其他需要强调的地方，小一些的字体可以用于页脚和辅助信息。谨慎使用粗体字和斜体字，保持页面整洁，阅读舒适。通过精心的页面规划与编排，使文字配合画面，达到页面的和谐与统一。

除字体和字号外，色彩以及文字块的图形都是文字设计的重要元素，都是调整信息读取的清晰度、易读度的因素。如为文字、文字链接、已访问链接和当前活动链接等选用不同颜色，使欲强调的部分更加引人注目。

总之，文字的主要功能是在视觉传达中向大众传达Web应用的各种信息，因此，需要提高文字的可读性、易读性以及表现的吸引力。

5）LOGO

LOGO（标志、徽标、商标）是一种具有象征性的大众传播符号，它以精炼的图形表达特定的含义，借助人们的符号识别、联想等思维能力，传达Web应用的文化内涵和内容定位，是Web应用对外宣传自身形象和主旨的工具，是最吸引人、最容易被人记住的标志。图6.2所示为幸福密码网的LOGO，展示了人的幸福从心而生，促使人们向好、向善和向上的主要理念。LOGO在每个版块中根据版块自身的风格、主色调而调整色调。



图6.2 幸福密码网的LOGO

LOGO的设计要在Web应用制作初期进行。从Web应用的长远角度出发，设计出能够长时间使用、最能代表该Web应用主旨和内涵的LOGO。LOGO的设计和创意来自于Web应用的名称和内容。

（1）Web应用有代表性的人物、动物、花草等，可以用它们作为设计的蓝本，加以卡通化和艺术化，如Disney的米老鼠、SOHU的卡通狐狸等。

（2）专业性的Web应用，可用本专业有代表的物品作为标志。比如中国银行的铜板标志，奔驰汽车的方向盘标志，等等。

（3）最常用和最简单的方式是用自己站点的名称作标志。采用不同的字体、字母或字的变形、字母的组合可以很容易制作好自己的标志。

好的LOGO应该具有可识别性、功用性、独特性、多样性、艺术性和内涵性，LOGO在体现Web应用自身的象征意义的同时，简单可识别、独特、多样并赏心悦目，在各级页面的页眉位置进行展示，而且也通常被设计成可以回到首页的超链接，达到好的用户体验。

6.3.5 美学设计

好的Web应用页面在高效显示其内容和功能的同时，在美学上也应当是令人愉悦的。美学设计是一种艺术，是对页面设计和内容设计技术方面的补充。没有美学，Web应用的功能再强大，却可能不吸引人；有了美学，Web应用就能够将用户吸引到一个以用户为核心的世界。

1）文字

上一节对页面元素文字的设计，强调字体、字号等，而文字的图形化强调它的美学效应，把文字作为图形元素来表现，同时又强化了原有的功能。将文字图形化、意象化，以更富创意的形式表达出深层的设计思想，能够克服页面的单调与平淡，从而打动人心。

2）色彩

色彩影响着人们的视觉注意力，能唤起人类的心灵感知，对人的视觉产生心理影响。它既是自然界最简单的事物，包括了红、橙、黄、绿、青、蓝、紫等七色，又美丽而丰富，五颜六色、千变万化。任何一种彩色都具有色相、饱和度和亮度3个属性。

Web应用给人的印象来自于视觉，不同的色彩搭配会产生不同的效果，并有可能影响到用户的情绪。在色环中，从红色到黄色的范围会使人感到温暖，主要包括红、橙、黄等色，称为暖色。由蓝绿到蓝紫范围的色会使人感到寒冷，主要包括蓝绿、青、蓝、蓝紫等色，称为冷色。而黄绿、紫等色属于不冷不暖的中性色。在Web页面中，为了营造出温馨祥和的气氛，多采用大面积的暖色来实现；而一些严谨、敏感的信息多以冷色表现。所以Web页面的色彩设计既要考虑Web应用主题，又要考虑用户体验，甚至还需考虑民族性（即环境、文化、传统等因素的影响，对于色彩的喜好也存在较大的差异），使页面具有深刻的艺术内涵，从而提升Web页面的文化美学品位。

Web页面中色彩总的应用原则应该是"总体协调，局部对比"，也就是说，整体色彩效果应该是和谐的，只有局部的、小范围的地方有一些强烈色彩的对比变化。考虑到Web页面的适应性，应尽量使用Web页面安全色。

色彩的搭配好坏决定了一个Web应用展示色调的好坏，色彩搭配一般遵循以下几个基本原则。

（1）选定一种主色调。可以根据Web应用的主题、服务对象和所要表达的内容和风格，分别设计不同的主色调。

（2）主题色和辅助色的搭配运用。主色调主导整个Web应用页面风格和意境，在和谐统一的基础上搭配少量辅助色以体现页面个性。

（3）把握对比与调和。各种色彩的色相、明度、纯度和面积、形状、位置以及心理刺激的差别构成了色彩之间的对比。将有差别、有对比效果的色彩经过调整和组合，调和成统一而和谐的整体效果。如运用同类色、邻近色、对比色，以及黑、灰和其他中性色，达到不同的效果。

（4）文字色彩、图形色彩与底色搭配合理。文字色彩与底色应保持较强的对比度（即可视度），以突出文字。对于图形感较强的页面，要将底色与图形色的反差拉开，起到烘托主题图形和页面色彩主次分明的作用，并结合图形的面积处理底色和图形色：图形较大，则使用主色调；图形小，则可配置鲜艳的色彩。

(5) 适当使用中性色。中性色在应用于背景色时可以增强暖色的效果。

(6) 适当使用流行色。流行色是一种与时俱进的颜色,其特点是流行快而周期短。当页面面向的对象是一些前卫人群时,或者页面宣传的商品是流行性的服饰、饰品等,可以根据消费者求新求奇的心理,采用当前流行的颜色作为色彩搭配的主要依据。

色彩是时代的代表,在以 Web 2.0 为主导的页面设计时代,为了让页面更加生动,更符合以人为本的理念,世界许多设计大师领导了一些色彩流行趋势,具有代表性的有以下两种:以水蓝、淡黄、粉红、黄绿、灰白为代表的浅色系,这类颜色色彩柔和,让人感觉舒服,代表站点有 http://www.gmail.com;另一类是高饱和度的色彩,此类色彩直观,有冲击力,代表站点有 http://www.rollyo.com。

3) 风格与创意

风格是艺术作品在整体上呈现出的具有代表性的独特面貌。Web 页面设计虽然不是一种纯艺术的活动,但是,为了达到好的用户体验,它也要求设计要具有鲜明的个性和特色,给人以美感。

Web 应用的风格是其整体形象给用户的综合感受,应该体现与众不同的特色,包括 Web 页面字里行间透露出作者或企业的文化品位和行事风格。对于不同性质的行业,就像穿着打扮应依不同的性别以及年龄层次而异一样,应体现出不同的风格。例如,政府部门的风格一般比较庄重;而娱乐行业则可以活泼生动一些;文化教育部门的风格应该高雅大方;而商务部门的风格可以贴近民俗,使大众喜闻乐见。

风格包括 Web 应用的版面布局、色彩、字体和浏览方式等。在有风格的 Web 应用上可以获得除了内容之外的更感性的认识,比如 Web 应用的品味、对用户的态度等,甚至可以感受到这个 Web 应用是粗犷豪放,还是清新秀丽,抑或是温文儒雅、执着热情,例如,Disney 生动活泼,IBM 严肃庄重,幸福密码网明亮温馨。

风格的形成并非一次到位,可以在实践中不断强化、调整和改进,如将站点 LOGO 尽可能地放在每个页面最突出的位置,突出标准色彩,使用标准字体,使用统一的图片处理效果。

创意是 Web 应用生存的关键,是一种灵感,一种思考的结果,是传达信息的特殊方式,可以从良好的策划定位、页面设计新颖化、内容新颖化等方面得到体现。好的创意会使 Web 应用更生动、更具吸引力、更有思想。将网络虚拟环境与现实结合起来往往能产生奇思妙想,如在线医院、电子社区、电子杂志、在线咨询大厅等。

6.3.6 展示设计原则

Web 页面的展示设计是一项既简单又复杂的工作,没有严格一致的规范,但是有一定的原则可以遵循,使页面达到合理并具有美感,还要把握其特殊性。说其简单,是因为设计人员只要遵守一些基本原则就行,那就是设计人员的设计应该增强而不是减弱 Web 页面所要提供的信息。说其复杂,是因为 Web 页面设计要考虑到很多方面的问题,需要有灵感、技巧以及美学知识。

无论使用何种手法对画面中的元素进行组合,设计者都一定要遵循主题鲜明、统一、平衡、连贯、分割、对比、和谐、留白和无障碍设计几大基本原则。

(1) 主题鲜明:Web 页面设计的最终目的是达到最佳的主题效果,要取得这种效果,一方面通过对 Web 页面主题思想运用逻辑规律进行条理性处理,使之符合用户获取信息的心

理需求和逻辑方式，达到快速理解和吸收；另一方面通过对 Web 页面构成元素运用美学原理进行条理化，更好地营造符合设计目的的视觉环境，突出主题，增强用户对 Web 页面的注意力，增进对 Web 页面内容的理解。只有这两个方面有机地统一，才能实现最佳的主题诉求效果。

(2) 统一：指设计 Web 页面所展现的内容与页面的结构、风格、板式、设计语言等表现形式之间的整体性、一致性。好的设计必定是形式与内容的完美统一。

(3) 平衡：指以页面中心点为支点，页面的上下左右的分量上应该给人以匀称的感觉，即左右平衡与上下平衡。实现页面平衡的方式有对称平衡、非对称平衡和辐射平衡。对称是最常见、最自然的平衡方式，给人以庄重和稳定的感觉，通常用来设计比较正式的页面。非对称平衡体现更多的视觉乐趣，如果把握不好页面就会显得乱，因此使用起来要慎重。辐射平衡是页面中的元素以某一个点为中心展开而构成的。

(4) 连贯：指要注意 Web 页面之间的相互关系。在 Web 应用设计中设计人员应利用各组成部分在内容上的内在联系和表现形式上的相互呼应，并注意整个页面设计风格的一致性，实现视觉上和心理上的连贯，使整个页面设计的各个部分相互融洽，形成完美的整体。

(5) 分割：指将 Web 页面分割成块，每块之间存在视觉上的不同，以使用户一目了然。在信息量很多的时候，为了使用户能够将各种内容区分开来，就要注意将 Web 页面进行有效分割。分割不仅是表现形式的需要，而且是对于页面内容的一种分类归纳。分割一般分为两种：横向分割和纵向分割。

(6) 对比：指通过矛盾和冲突，突出强调某些内容，使内容更易于辨认和接受，使设计更富生气。常用对比实现方法是使用大小、颜色、字体、重心、形状、纹理等。例如，内容提要和正文使用不同颜色，已访问和未访问的链接使用不同颜色，即颜色对比；大标题小正文，即大小对比；还可以使用图像体现对比，题头的图像明确地向用户传达本页的主题。设计人员在使用对比时应慎重，对比过强容易破坏美感，影响统一。

(7) 和谐：指整个 Web 页面符合美学原理，浑然一体，既要看结构形式，又要看作品所形成的视觉效果与人的视觉感受，达到沟通和共鸣。

(8) 留白：指的是在 Web 页面留出一些空白，让用户有更大的想象空间。上上下下、里里外外都塞得满满当当的 Web 页面会带给用户拥挤、烦躁、混乱的感觉。在页面的段落、图片、按钮和其他元素之间留白，可以使整个页面上的元素排版的松紧呈现出高低错落，跌宕起伏，这样浏览起来就会富有节奏感，像是在听一首好听的歌曲。还可以通过减少条目之间的空间和增加它们与页面中的其他条目之间的空间来进行分组。留白需要注意：元素之间的留白不能太大，这是基本的原则，留白过多，页面会变得零碎；文本中间的间隔不能太大，文本应当紧凑，尤其汉字文本，如果字与字之间或者行与行之间空白太大页面就会非常难看。

(9) 无障碍设计：指设计时考虑身体或心理有障碍人员，以满足他们的身心需要。例如，针对视觉障碍(如色盲)，可以在 Web 页面设计时嵌入使用屏幕朗读或页面颜色的搭配；考虑听觉有障碍的人群，可以考虑在 Web 页面设计中对于重要的语音信息，同时给出相关文字提示信息，等等。

以上提到的内容尽管是 Web 应用设计中需要遵循的一些指导原则，在实际设计时根据实际情况进行灵活应用才能做好 Web 应用的展示设计。

另外，针对 Web 页面的浏览速度和页面的适应性，在进行页面设计时，需要注意如下一些原则。

（1）源代码优化。良好的编码习惯和正确而精简的代码能够形成优美的 Web 页面文件，并减轻了工作量，提高 Web 页面的性能，因此要做到少用特效，文件重用，清除无用代码。

（2）图像优化。图像优化的目的是在美化的同时尽量减小图像大小以提高 Web 页面的下载速度。因此，可以采用合适的图像格式；说明图像大小如＜IMG SRC＝"a.GIF" height＝310 width＝320＞，使浏览器会预留下图像的空间而先显示文本信息；缩小图像文件，可使用缩略图，缩略图是一种很小的图像。

（3）表格优化。由于浏览器在载入 Web 页面时，读完整个表格才将它显示出来，因此，在设计页面表格时，可以设定表格的宽度、高度、边框、背景色、对齐方式等参数，并尽量避免过多表格嵌套层次和将所有元素嵌套在一个表格里。

（4）搜索引擎优化（SEO）。SEO 是通过优化 Web 应用结构、Web 页面代码和内容等方式，从而使得搜索引擎更易于找到 Web 应用内的页面，提高 Web 应用在搜索结果中的自然排名。（详细描述参见第 9 章）

（5）目录结构清晰。Web 应用的目录优化是指构建 Web 应用时创建清晰的物理和逻辑目录结构，来确保搜索引擎和用户的访问。Web 应用的物理结构目录及所包含文件命名要规范，结构清晰，而逻辑链接结构要控制好层级并设计好整个应用内的互连。

（6）避免使用框架。框架会增加浏览器对服务器的访问连接请求数而致使服务器负担加重。因此，应尽可能避免使用框架（包括 IFrame、Frame）。

（7）文字优化。使用 CSS 样式表指定文字的样式和颜色，原则是以清晰且与整个页面搭配和谐为准。

Web 应用的页面设计需要综合考虑上述各指导原则，以提高 Web 应用的可用性、可访问性以及性能（详见第 11 章）。同时考虑 Web 应用的多渠道访问特性，设计时体现设备无关性，使 Web 应用的页面能够感知不同上下文环境，自适应地展现适合当前访问应用的设备的页面。

6.4 内容设计

对于大多数 Web 应用而言，"内容是王"。在建模应用期间构建了内容模型（本书利用 UWE 进行建模），静态内容模型用 UML 类图表示，动态内容模型用状态图表示，但是并没有考虑这些内容在 Web 应用中的安排部署方式。内容设计主要考虑提供哪些可用内容（组件）及其组成和导航（网），其目标是把通常以一组详细的内容对象表示的内容需求转化为 Web 应用具体的信息设计，关注如何组织、访问和管理这些内容，定义所有内容的布局、结构和作为 Web 应用的一部分进行展示的内容大纲，并建立内容对象之间的关系。

Web 应用的内容是用户通过 Web 应用获取得到的信息，其结构（即内容的组成）是为用户提供的内容视图，其行为是为用户提供的访问视图的方式。随着 Web 应用技术的发展，Web 应用信息设计逐步走向成熟，可分为抽象程度高的信息架构设计和访问信息的特定导航结构设计。

6.4.1 信息设计方法

信息设计的总体策略通常会结合自底向上和自顶向下两种方法。

1) 自底向上的信息设计

自底向上的信息设计一般应用于小型的Web应用,仅涉及实际Web页面的设计和构建,并逐步把这些Web页面链接起来,这样Web应用的信息视图和结构就能够有机联系起来一同出现。然而,对于大型Web应用来说,自底向上方法产生的解决方案会阻碍用户定位信息的意图,表示信息时断章取义,并且限制对变化的可适应性。

2) 自顶向下的信息设计

自顶向下的信息设计强调Web应用内主要内容种类的总体组织、相互关系和结构。这种高层设计方法针对Web应用的总体结构、组织信息的方式以及用户可能的访问信息的方法等方面,这是信息架构的范畴。

6.4.2 信息架构

信息架构(Information Architecture,IA)是信息空间结构的高层设计,有助于任务的完成和对内容的直观访问,涉及组织、标识、导航和搜索的设计,目的是帮助人们在网络和Web环境中更成功地发现和管理信息,有效地解决用户的信息需求。信息架构的主要任务是对信息进行分类、归类,以及组织。信息架构具有如下特性。

(1) 与多种动态数据进行组合。架构必须支持把不同的信息项归为一种表示并表达这些项之间的限制能力。例如,新闻系统中页面中的视频是否需要在新窗口中播放,是否自动在播放完之后重播,等等。

(2) 高层表示的规格说明。架构应该能够指明多个信息项之间的约束。例如,可能想指明某种图片种类(如摄影作品等)在没有包含摄影者名字的情况下是绝不应该被展示的。在新闻系统中,用户的姓名和登录状态都一直显示,以树立用户对应用的信心。

(3) 时间关系。有些信息项可能与时间有关系,这对其显示可能非常重要。例如,到一个事件信息的链接可能只有在事件被挂起时才是可用的。

(4) 链接的上下文和链接语义。很多Web应用,尤其是那些支持适应性和个性化的Web应用,一个重要特征是能够根据选择的链接来控制显示,"如果用户沿着链接A,那么结果将显示在一个新窗口中;如果用户沿着链接B,那么用结果取代现有信息。"

(5) 内容与信息分离。内容是可用数据源的集合,信息是对Web应用用户有用的那些内容,好的信息架构应该加以区分。

(6) 信息和应用分离。Web应用的信息架构应当区分用户可能发现的有意义的信息和这些信息可能被放置和访问的结构方式。例如,新闻系统中的新闻分类应该是独立模型,而非直接作为新闻信息一部分。

(7) 应用和展示的分离。把展示机制和应用分离有助于提高Web应用的可移植性和通用性。例如,新闻的格式可以通过显示模板展示。

6.4.3 组织内容

就算不考虑 Web 应用的界面和美学复杂性，最起码信息要可获取。如果内容的内部结构混乱，那么 Web 工程开发团队要完成设计、导航和搜索机制，以实现为用户提供其所需信息就非常困难。

组织内容的目的之一类似于图书馆中书的组织以及书内页面的组织，使用户可以找到他们需要的内容。图书馆使用那些已经演化了几百年的信息分类(结构)，而一个特定 Web 应用需要设计独特的定制结构。这些结构应该直观，并且应该考虑用户最可能希望的信息组织方式。有些情况下，这种组织是信息本身所固有的，如在线电话簿中信息的自然排序。另外一些情况下，采用可以提供自然组织的隐喻，使用户一旦了解了这个隐喻(如一个购物车)，信息结构就很容易理解。对信息的组织、标注和叙述的方式将会影响用户理解这些信息的方式，因此，要为用户易于找到其问题的正确答案而组织信息。

组织内容的其他目的如商品推荐。例如，电子商务应用可能把信息结构设计为能使用户容易看见那些鼓励用户购买更多的信息，如 amazon. com 内用户个性化功能，确保根据 Amazon 的每个顾客之前的导航和购买，展示给他们特定的图书和相关产品。在新闻系统中，对内容进行适应，当用户登录后，根据用户历史阅读、评论，提供有关类型的新闻内容推荐。

根据不同 Web 应用的不同需求，将信息的组织方式分为线性结构、层次结构、网络结构、矩阵结构，如图 6.3 所示。

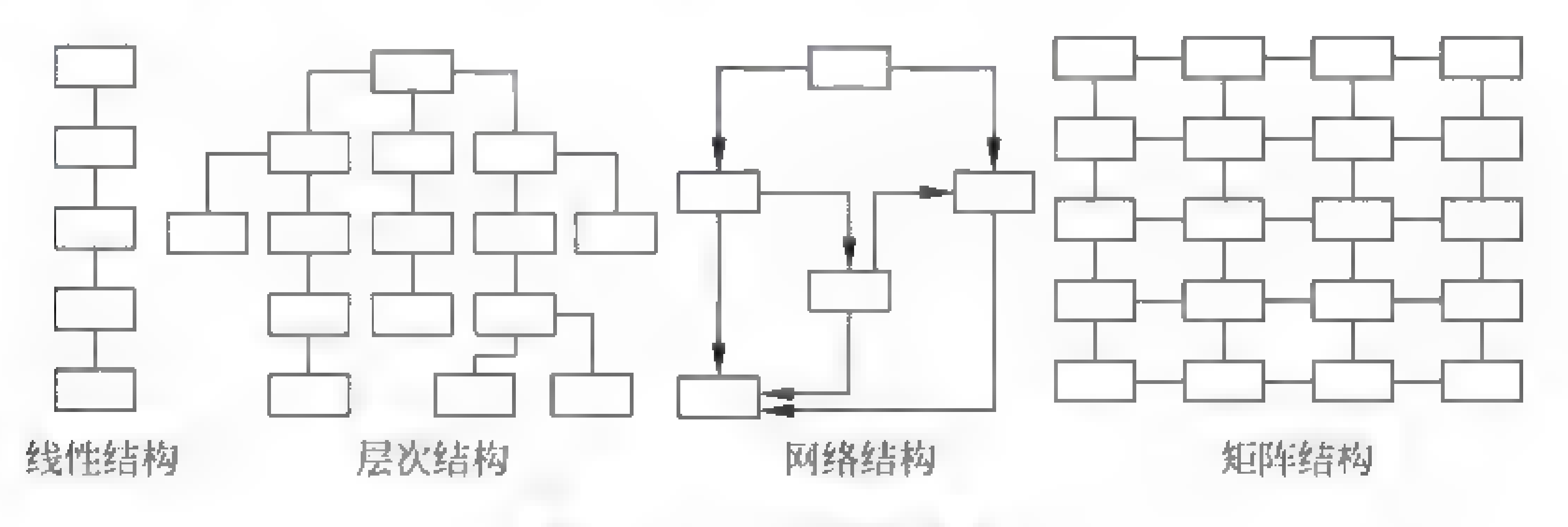


图 6.3 信息结构示意图

线性结构可以用于保存一个原始文档的顺序结构或用于控制访问，即当可以预测交互顺序性并很常见时，选择线性结构。例如，在线培训中，学生通常通过顺序的方式进行学习访问。当内容和处理变得越来越复杂时，线性结构就需要加上更复杂的分支结构，其中，可以触发或转换可替换的内容，来获取补充内容。

矩阵(或网格)结构是当 Web 应用的内容按类别组织成两维(或更多维)时采用的层次化结构，适用于内容规则性很高的情况。矩阵的水平方向分成不同问题信息，矩阵的垂直方向分成描述症状、原因、解决方案、工具和过程等信息。当用户要解决一个具体问题时，可以对网格进行水平导航，然后进行垂直导航来理解问题的性质。

层次结构毫无疑问是最常用的 Web 应用结构，用于反映自然信息分类法。例如产品目录适合使用层次结构。在很多情况下，多个不同的层次可能都合适。

网络(或图)结构由把信息空间中常见的或相关的概念绑定在一起的关联链接组成，在

Web应用内交叉连接相关方面非常有效，并且提供给用户灵活的导航，通常作为层次结构之上的层来使用。网络结构使用得当，有助于用户进行有效的浏览；过度使用，用户将会被选择所迷惑或产生他们在哪里以及哪些信息可用的感觉。例如 www.wikipedia.com 的内容有一些分类，其基础是内容之间具有大量的交叉引用，而非自然层次。

上述几种结构应综合运用，如 Web 应用的总体结构是层次结构，但是部分结构展示出线性特征，而结构的另一部分是网络结构，使 Web 应用的结构、将要展示的内容以及将要进行的处理相匹配，达到既满足用户的信息要求，又易于导航。例如层次结构，窄而深使得在任何页面可选择的链接少，导航选择简单，但是导航路径要长得多；宽而浅的导航路径短，但是在页面内选择就复杂得多。因此，要权衡层次结构的广度和深度，采用一些设计指导原则，如 7+2 规则结合 Web 应用的选项复杂性和选择之间的相异程度相关的扇出，避免层次深度超过 3～4 层，种类有明显区分，等等。

有了内容组织就可以为信息架构补充更详细的设计信息，如内容的动态和静态特性、用户个性化支持、与 Web 页面的映射、导航路径等。若采用自上而下的方法进行开发，则从主页面开始，然后逐步增加附属页面；若采用自底向上进行开发，则考虑内容对象，并审查它们如何被聚类（对层次结构而言）或排序（对线性结构而言）。

6.4.4 访问信息

Web 应用中的信息除了信息结构外，信息的访问还受导航机制及其特征等因素的影响。通常，这些影响因素有：①使用户在任何给定的时间知道哪些导航选项是可用的机制；②为用户提供说明他们在哪里以及他们正在看什么的界面机制；③允许用户在信息结构内游走的导航机制。

导航机制决定使用者如何浏览和检索分类内容，像现实中的路标或地图一样，它有助于用户了解自己所处位置以及从这个 Web 应用中能获得什么。其中多种导航元素嵌入在页面中，为用户浏览整个 Web 应用提供了导航路径。导航机制分为全局导航、局部导航、语境导航和补充导航 4 类。

全局导航系统是对整个 Web 应用内容进行导航，提供到达 Web 应用内任意页面的链接导航，引导用户对整个 Web 应用内容做纵、横向浏览，使用户能够深入 Web 应用的主要内容域，并可从 Web 应用各处返回到主页。

局部导航作为全局导航的补充，为全局导航的下层内容进行细分而设，主要目的是让用户在一个内容域里按层次浏览信息。当 Web 应用栏目内容比较复杂，每个栏目的具体内容还需进一步分类组织时，就需要设置局部导航。

语境导航主要是让用户浏览上下文之间相关的内容，其目的是提供语境提示，引导用户阅读与当前内容相关的信息。可以是嵌入文本段落中的文字型链接，也可以是一种位置导航，多用于非关键性信息。

一般导航都是伴随着页面内容而存在的，而补充导航则是以在页面内容之外的方式向用户提供进入内容的途径，提供了专门入口，而不必经过原始层次结构依次查询。这些导航元素是保证 Web 应用的可用性和易访问性的关键因素，其类型有站点地图、站点索引、指南、目录、下拉菜单等。

帮助用户理解自己所处的上下文和正在阅读的信息，以及在信息结构中游走的一些指

导原则如下。

(1) 明确标记。开发所有链接锚点清晰、明确且一致的标签,准确描述链接的目的地,并且确保用户通过链接访问时能够知道自己已经访问过哪里。

(2) 全局链接。在每个 Web 页面上都有一些链接到经常访问的 Web 应用位置或功能,使用户可以随时跳到那些位置,而不需要返回到首页或顺着其他规定的导航路径,例如,首页、帮助、联系、网站地图、搜索、新闻、关于、注册和登录等。

(3) 快捷方式。绕过常规导航路线,直接到达信息空间内的一个特定位置。例如,使用子菜单来使用户绕过中间页面。

(4) 层级(Breadcrumbs)和轨迹。提供包含在每个 Web 页面中的导航路径描述,描述当前页面在信息结构内部的位置。当导航到信息架构深层时,能知道来自哪里。同时,将浏览轨迹显示为可激活的链接,可以使用户快速返回到导航路径上的任意一点。

(5) 个性化。每个 Web 页面都应该明确表明 Web 应用的性质或显示的信息所属应用,同时应该提供对上下文和 Web 应用意图的指示,还应当提供到首页的链接。

(6) 搜索机制。允许用户绕过导航结构而直接跳到 Web 应用内的特定位置,因此,常常是那些"现在"就需要信息的用户的首选。

上述原则需要综合使用。例如,在使用搜索机制时,可能会破坏导航结构上的逻辑顺序要求,匹配结果量太大导致用户迷惑和迷失,而层级可确保信息空间中所有潜在目的地的位置上都包含清楚的信息,是完成这一目标的有效机制。

6.5 功能设计

Web 应用功能设计主要关注 Web 应用所支持的行为和功能,包括如工作流支持、内容和界面自适应和(或)定制、订单录入、数据库处理、计算功能等。功能设计从用户功能设计开始,作为 Web 应用功能建模的延续,以交互模型和功能模型来描述,如序列图、状态图和活动图等。而且功能设计和信息设计并行交织进行,两者之间相互依赖。

和早期 Web 应用相比,现在的 Web 应用功能设计比传统应用的功能设计复杂得多。设计者必须考虑由 Web 基础架构本身所赋予的一些必不可少的约束,如分布式模型(使得信息处理和用户响应变得复杂)、安全问题、从 Web 浏览器继承来的受限的接口模型。同时,如果相关信息架构比较复杂,那么想要做到有效的集成就会更困难。

在 Web 应用设计时,除了进行传统软件类似特性的设计之外,还需要考虑技术对 Web 应用开发的影响,必须衡量所采用的方式,应用需要具有可扩充性、可伸缩性和可维护性。要达到平衡,最大困难是组件之间的相互作用。和新闻发送器类似的 Web 应用通常不需要事务支持,而在线购物则必须反映多种产品阶段,如从订购到修复,需要事务和工作流支持,以及遗留数据库和软件集成。

本节主要讨论集成和跨企业分布式 Web 应用设计时,技术对设计产生的影响。

6.5.1 集成

集成可以从三个功能层次进行:数据层、应用层和过程层。在数据层进行集成时,要保

证不同应用的数据之间可以进行转换和复制。例如，从一个应用的数据库导出数据再导入到另一个应用的数据库中，或者通过JDBC连接数据库，这种方式不涉及Web应用本身，也无须我们验证数据。应用层集成的交互通过API进行，也就是时间和语义密切交织，许多细节取决于中间件。在过程层集成要根据基础架构单独对业务模型进行建模，通常被认为是最高层集成。

中间件作为连接应用的技术，不同方法的复杂性和其目标区别很大，如IPC、RPC、EBC、MOM和一些分布对象方法。基于XML的方法逐步成为Internet上的通用语言，XML不仅能够更好地描述半结构化数据，而且描述很多分布式应用标准，如简单对象访问协议(Simple Object Access Protocol，SOAP)，Web服务描述语言(Web Service Description Language，WSDL)，统一描述、发现和集成协议(Universal Description，Discovery，and Integration，UDDI)。SOAP处理不同Internet协议上的消息和调用，如HTTP、SMTP等协议；WSDL描述Web服务的接口和如何访问Web服务；UDDI提供类似于数据库的功能支持发布和搜索Web服务。

6.5.2 分布式Web应用

在Web应用的软件特性实现方面，分布式的地位越来越重要。分布式软件正如链接到远程Web页面一样，从分布式软件访问远程Web应用，可以看做服务与服务交互。服务通过定义好的接口方式提供。例如，eBay不仅有单点认证，还支持微软的Passport，Google可以在外部应用中通过SOAP集成其搜索引擎。

这种粗粒度开放组件市场对Web应用设计具有严重的影响。外部开发的功能可以降低开发成本，组件或服务的质量可能好一些，但是对这些服务的控制代价却可能很高。例如，微软Passport的安全漏洞使最初的热情减退，对安全性要求高的应用中对外部服务的接受门槛相应也会非常高。基于组件的方法虽然成本不会低，但因其高可重用性而有助于开发出高质量的应用，有助于树立用户的信心。

Web应用在B2B方法的基础上，逐步发展成大型分布式系统。这种集成不仅涉及企业内部应用，还涉及到第三方应用或服务。有些企业已经通过供应链管理(SCM)扩展访问第三方应用。Web服务旨在标准化Web上的集成访问方式，建立在XML和SOAP、WSDL、UDDI、BPEL等协议之上，处理企业和公司之间的相互交互，支持服务事务和业务过程协作，等等。

6.5.3 功能设计原则

不管是复杂的分布式Web应用，还是简单的单独应用，不管是否进行集成，从资源角度来讲，必须充分利用企业已有的软、硬件及网络资源，以需求为导向，切合实际，操作简便，达到“实用、简单、好用、耐用”的要求。

Web应用的功能设计在Web应用的构建中起到很重要的作用。设计出新颖强大的功能，对于Web应用的构建和推广营销来说是一个关键的环节。Web应用功能性的设计，需要考虑Web应用功能的先进性、可用性、实用性和可扩展性等一些常用原则。

1) 先进性原则

立足于高起点、高标准、高效益，在切合实际的前提下采用国际上先进、成熟的技术，使

整个 Web 应用从整体上达到先进、灵活、高效的目标。既健壮，可扩展性又好，既能保证满足目前的需要，又能保证系统能够适应企业今后快速发展的需要。

2）可用性原则

Web 应用设计出新的功能的确是一件让人感到兴奋的事情。太多的 Web 应用设计者看到新功能的好处，却忽视了整个 Web 应用的可用性。每增加一个新的功能，会给已有功能的使用带来影响，产生用户适应新功能的代价，降低整个系统的可用性，导致用户流失。而且，如果 Web 应用的功能只有少数几个人会用的话，或者说看起来很强大，实际用起来很繁杂的话，用户一般会觉得无所适从，很少会有耐心去学习和摸索新的功能，再漂亮再强大的功能也会让人觉得望尘莫及。因而，Web 应用在设计开发新功能或是 Web 应用改版时，要循序渐进，要尽量简单化，要让用户感觉不到变化。

3）实用性原则

Web 应用拥有丰富的功能证明了 Web 应用的强大，但这个强大并不是其功能越多越好，很多时候复杂的功能设计只会影响 Web 应用整体。

类似特定领域建模工具应该针对适用领域一样，Web 应用功能设计应该遵循实用性。实用性的功能是对于用户使用该 Web 应用有实质性意义的功能，是为 Web 应用的核心目标服务的功能。与 Web 应用目标关系不大的功能堆积得越多，实用性越差。功能是为了完成 Web 应用的目标，如果功能与 Web 应用核心目标之间没有多大关系，即使这样的功能获得了较多的使用者，并为 Web 应用带来一定的流量，但对 Web 应用的总体目标意义也并不是很大。

4）可扩展性原则

正如"在建造船舶的时候，如果是按照小舟的模式来设计的话，建造出来的船只能在小河上行驶；如果是按照军舰的模式来设计的话，将来这艘船有可能乘风破浪，越洋航行。"在设计 Web 应用的功能时，如果给它套好了条条框框，这样只会局限其发展与扩展。而在设计时为其发展留下可扩展的空间，就可根据市场的变化对功能进行不断的扩充与完善。

Web 应用的功能设计作为构建中最最核心的一步，设计者在选择的时候要慎重，更要以长远的眼光来看待 Web 应用的发展。

6.6 总结与展望

无论 Web 应用系统架构师设计的架构有多好，开发人员的技术水平有多高，毕竟他们都不是系统的最终用户，只有最大限度地满足客户的需要才是 Web 应用的立足之本。所以，有效的 Web 应用设计必须做到以用户为中心，进行人性化、智能化的设计，需要时刻注意到 Web 应用的可用性、性能和安全性等内容，把 Web 应用设计成易于导航、操作方便、使用快捷的用户满意的系统。

本章把 Web 应用设计分为交互设计、展示设计、信息设计和功能设计，这几方面设计有机结合，采用可扩展的适应性好的技术，最终展现在用户面前的是一个功能完备、内容充实、信息完整、布局合理、色彩分明、协调美观的界面。良好的 Web 应用页面设计是一个好的 Web 应用的必须条件，其好坏直接关系着用户对 Web 应用的满意程度，所以在 Web 工程中需要特别强调与注意。

Web应用设计的一个亟待发展的方面是上下文感知和设备无关性趋势。随着Web技术和标准的发展，既要适应越来越普及的基于位置等上下文的服务，比如饭店的Web应用基于用户的位置提供相应的特色信息，又要适应越来越多种类的访问Web应用的设备，尤其是移动设备，它将是Web应用的主要访问设备之一。一方面，采用平台无关的协议描述并结合Web服务，采用GPS等提供移动位置支持等；另一方面，尽量提供满足最小设备的应用展现，基于CC PP(DIWG，W3c 2001c)标准提供上下文描述并适合各种不同设备，或使用转码服务器进行应用的适应性改变。

Web应用设计的另一个亟待发展的方面是重用性，主要表现在3个概念方面的发展：一是网由元素和链接组成，已不足以表达Web应用的软件和信息设计，需要特殊的聚合概念；二是需要进一步设计出统一的设计符号；三是新的组合概念，如基于事件的通信以及多维链接的超文本概念。

第7章 Web应用构建与部署

定义了Web应用需求，选择了架构并进行了适量的设计后，即澄清了"什么"，就进入了构建阶段，即"如何"实现这些设计。构建阶段包括一系列的选择、编码、内容创建、集成、重构以及测试(由于测试的重要性故单独在第8章介绍)活动，以构建出可以部署并交付给最终用户使用的Web应用。选择什么样的实现技术，是Web应用成功的一个关键因素，最基本的原则是从内容、表示、功能分开考虑，而且需要考虑架构中分布式和与其他系统之间的交互，以及细粒度增量的部署。因此，需要了解各种技术的特点以及这些技术适用于什么样的架构。

Web应用的实现技术与传统软件实现技术相比，最大的特性来自于Web标准。因此，本章从请求、响应和通信三个角度来考虑Web应用的实现技术，介绍Web应用常用的一些通信协议、客户端实现技术和服务器端实现技术；并简要说明部署活动及其环境。

7.1 Web应用构建原则

从技术层面看，Web应用架构的核心：用超文本技术(HTML)实现信息与信息的链接，用统一资源定位技术(URL)实现全球信息的精确定位，用应用层协议(HTTP)实现分布式的信息共享。

Web应用的构建是使用Web应用开发工具和技术来构建已经模型化的Web应用。要充分发挥Web应用的内在潜力，挖掘应用深度和扩大适应能力，需要采用先进的应用架构和实用为本的原则，使得系统既能满足业务需求，又能适应未来发展的需要。因此，在构建Web应用时需要尽量遵循如下一些原则，以指导构建活动。

(1) 准备原则。在创建一个简单的Web页面或写一行代码之前，需要做到以下几点。

① 理解正要解决的问题。

② 理解基本的Web应用设计原则和概念。

③ 挑选一种适合要构建的组件的语言和组件运行环境。

④ 选择一个提供可以简化工作的工具的环境。

⑤ 构建在组件完成时需要用到的一组单元测试。

(2) 选择原则。当选择已有的可重用组件和对象时，需要做到以下几点。

① 考虑技术环境的限制。

② 使组件与信息和功能的环境相匹配。

③ 考虑开发人员和维护人员的技能和知识。

④ 考虑知识产权问题、组件的所有权及其可移植性。

(3) 编码原则。当开始写代码时,需要做到以下几点。

① 编写自记录的代码。

② 遵守编程风格来约束算法。

③ 选择符合设计需求的数据结构。

④ 理解功能架构,构建相符合的接口。

⑤ 条件逻辑尽可能简单,并确保是可测试的。

⑥ 采用可读性好的编码风格,例如选择有意义的标识符命名和遵循其他的局部编码标准。

(4) 内容管理。要管理内容,需要做到以下几点。

① 选择符合设计需要的数据结构。

② 理解信息架构并创建与之一致的内容和导航结构。

③ 确保格式和数据结构的一致性。

④ 避免依赖私有数据格式。

⑤ 把内容看成可发行资料而不是软件。

(5) 创作原则。当创建 Web 页面(或页面模板)时,需要做到以下几点。

① 不断地考虑可用性问题。

② 记住解决可访问性问题。

③ 理解用户如何与 Web 应用交互,而不是希望他们如何交互。

④ 向竞争者学习。

(6) 集成原则。当集成组件和对象时,需要做到以下几点。

① 保留备份,以便能够把 Web 应用恢复到早期的版本。

② 找出不匹配或不一致的组件接口。

③ 及时识别需要重构的组件。

(7) 重构原则。当重构 Web 应用时,必须做到以下几点。

① 理解一般的重构。

② 当有重构机会时,即使是一点点也要进行,但不要进行不必要的变化。

③ 确保实现和设计以明显的方式进行沟通。

(8) 测试原则。在完成第一个组件之后,必须做到以下几点。

① 进行走查。

② 执行单元测试,更正发现的错误。

③ 选择那些最可能发现错误而不是隐藏错误的测试。

一组基本的原则和概念可以使开发团队开发出可维护的、可测试的 Web 应用(有关 Web 应用的测试将在第 8 章详细描述),应该在构建活动中时刻谨记指导构建活动的这些原则与概念。

7.2 Web 应用通信协议

Web 上计算机之间通过特定的通信协议完成通信。通信协议是网络上所有设备(如 Web 服务器、计算机及交换机、路由器、防火墙等)之间通信规则的集合,规定了通信时信息必须采用的格式及其含义。Web 应用通信协议是 Web 应用开发的基础,常用的 Web 应用通信协议有 HTTP、MIME、RTP/RSTP 和 MMS 等协议。

1. HTTP 协议

HTTP(HyperText Transfer Protocol,超文本传输协议)是基本的 Web 通信协议,属于应用层的面向对象的协议,是一个基于文本的无状态协议,是运行在 TCP/IP 上的网络应用层协议,是客户端浏览器与 Web 服务器之间的应用层通信协议。其基本功能是完成 Web 应用中必要的文件传输和基于 Web 的动态交互应用。

HTTP 是 Web 上的图像、图形、音频、视频、超文本信息的传输载体,它不仅可以应用于 Web 访问,也可以应用于其他 Internet 或(和)Intranet 应用系统之间的通信,从而实现各类应用资源超媒体访问的集成。

HTTP 协议的主要特点如下。

① 支持客户/服务器模式。Web 内容位于 Web 服务器,通过 HTTP 协议发布信息,存储并提供给 HTTP 客户端所请求数据。

② 简单快速。客户向服务器请求服务时,只需传送请求方法和路径。常用的请求方法有 GET、HEAD、POST,每种方法规定了客户与服务器交互的不同类型。由于 HTTP 协议简单,使得 HTTP 服务器的程序规模比较小,因而通信速度快。

③ 资源灵活多样。HTTP 允许传输任意类型的 Web 资源对象,如 Web 服务器上文件系统中的文本文件、HTML 文件、Word 文件、Adobe Acrobat 文件、JPEG 图形文件或 AVI 电影文件等静态文件,产生所需监控图像、股市数据或在线购物等内容的软件程序等动态资源。资源通过 URI(统一资源标识)进行唯一标识,通过 URL(统一资源定位)进行定位,如新闻系统主页 http: mynews index.htm。正在传输的类型由 Content-Type 加以标记。

④ 无连接。无连接是指限制每次连接只处理一个请求。服务器处理完客户的请求,并收到客户的应答后,即断开连接。

⑤ 无状态。无状态是指协议对于事务处理没有记忆能力,缺少状态意味着如果后续处理需要前面的信息,则它必须重传,这样可能导致每次连接传送的数据量增大。另一方面,在服务器不需要先前信息时其应答较快。

⑥ 双向传输。浏览器请求 Web 页面的时候,服务器把页面副本传输给浏览器,并且 HTTP 也允许浏览器向服务器传输数据,如用户通过表单提交内容。

⑦ 支持高速缓存。HTTP 允许服务器控制是否能高速缓存页面、如何高速缓存页面以及页面的生命期,也允许浏览器强制页面请求绕过高速缓存,从拥有该页的服务器上得到新的副本。

⑧ 支持代理。HTTP 允许在浏览器到服务器之间路径上的机器作为代理服务器,将 Web 页面放入高速缓存并从中响应浏览器的请求。

HTTP无连接、无状态，客户和服务器之间的交互不保留客户状态信息。对于交互型Web应用而言，需要借助其他一些技术来区分并发用户以及同一用户的相关联的请求。一般情况下，通过session(会话)来定义特定用户和服务器之间一定时间范围的相关HTTP请求。对session进行跟踪实现用户状态通常采用URL重写和Cookies两种方法。

2. MIME协议

MIME(Multipurpose Internet Mail Extensions，多功能因特网邮件扩充服务)是一种多用途网际邮件扩充协议，用于描述数据的类型，根据相应类型的规定决定数据处理方式的规范，在1992年最早应用于电子邮件系统中，后来也被应用到浏览器。服务器通过说明将发送的多媒体数据的MIME类型将该多媒体数据的类型告诉浏览器，例如让浏览器知道收到的信息哪些是MP3文件等，Web应用中借用该概念实现内容协商机制。

3. RTP/RTSP协议

RTP(Real time Transport Protocol，实时传送协议)是一个网络传输协议，它是由IETF的多媒体传输工作小组于1996年在RFC 1889中公布的。其中详细说明了在Internet上传递音频和视频的标准数据包格式。RTP一开始被设计为一个多播协议，但后来被用在很多单播应用中。RTP协议常用于流媒体系统(配合RTCP协议)、视频会议和一键通(Push to Talk)系统(配合H.323或SIP)，使其成为IP电话产业的技术基础。RTP协议是建立在用户数据报协议上的，和RTP控制协议RTCP一起使用。

RTP本身并没有提供按时发送机制或其他服务质量(QoS)保证，而是依赖于底层服务实现。RTP并不保证传送或防止无序传送，也不确定底层网络的可靠性。RTP实行有序传送，RTP中的序列号允许接收方重组发送方的包序列，同时序列号也能用于决定适当的包位置，例如在视频解码中，就不需要顺序解码。

RTSP(Real Time Streaming Protocol，实时流传输协议)是TCP/IP协议体系中的一个应用层协议，定义了一对多应用程序如何有效地通过IP网络传送多媒体数据，用来控制音频或视频的实时发送，并允许同时控制多个流。RTSP在体系结构上位于RTP和RTCP之上，使用TCP或RTP完成数据传输。

RTSP的语法和运行与HTTP 1.1类似，但并不特别强调时间同步，比较能容忍网络延迟。HTTP与RTSP相比，HTTP传送HTML，而RTP传送的是多媒体数据。HTTP请求由客户机发出，服务器做出响应；使用RTSP时，客户机和服务器均可发出请求，即RTSP可以是双向的。

RTSP提供了一个可扩展框架，使实时数据，如音频与视频的受控、点播成为可能。数据源包括现场数据与存储在剪辑中的数据。该协议目的在于控制多个数据发送连接，为选择发送通道，如UDP、组播UDP与TCP，提供途径，并为选择基于RTP上的发送机制提供方法。RTSP允许同时多个流控制(Multicast)，除了可以降低服务器端的网络用量，更进而支持多方视频会议。因为与HTTP 1.1的运作方式相似，所以代理服务器<Proxy>的快取功能<Cache>也同样适用于RTSP，并因RTSP具有重新导向功能，可视实际负载情况来转换提供服务的服务器，以避免过大的负载集中于同一服务器而造成延迟。

4. MMS 协议

MMS(Microsoft Media Server Protocol，微软媒体服务器协议)是一种流媒体传送协议，是用来访问并流式接收 Windows Media 服务器中.asf 文件的一种协议。MMS 是连接 Windows Media 单播服务的默认方法，若用户在 Windows Media Player 中输入一个 URL 以连接内容，而不是通过超级链接访问内容，则他们必须使用 MMS 协议引用该流。MMS 的默认端口是 1755。现在除了 Windows Media 外，如 KMPlayer 等流行的播放器也支持该协议。

5. FTP 协议

FTP(File Transfer Protocol，文件传输协议)用于 Internet 上的控制文件的双向传输。使用户可以把自己的个人计算机与世界各地所有运行 FTP 协议的服务器相连，访问服务器上的大量程序和信息。

FTP 的主要作用是让用户连接上一个远程计算机(这些计算机上运行着 FTP 服务器程序)，查看远程计算机有哪些文件，然后把文件从远程计算机下载到本地计算机，或把本地计算机的文件上传到远程计算机。

6. SMTP 协议和 POP3 协议

SMTP(Simple Mail Transfer Protocol，简单邮件传输协议)是一种可靠且有效的电子邮件传输协议，是建立在 FTP 文件传输服务上的一种邮件服务，主要用于传输系统之间的邮件信息并提供与来信有关的通知。SMTP 设计基于以下通信模型：针对用户的邮件请求，发送 SMTP 与接收 SMTP 之间建立一个双向传送通道，接收 SMTP 可以是最终接收者也可以是中间传送者，即"SMTP 邮件中继"。SMTP 协议是基于 TCP 服务的应用层协议，用户直接使用的是用于编写和发送的客户端软件，而 SMTP 服务器运行在远程的 Web 应用上。

POP3(Post Office Protocol V3，邮局协议版本 3)用于电子邮件的接收，它使用 TCP 的 110 端口。该协议规定了与 POP3 服务器进行对话的一系列命令和过程标准，它是因特网电子邮件的第一个离线协议标准。POP3 协议允许用户从服务器上把邮件存储到本地主机上，同时根据客户端的操作删除或保存在邮件服务器上的邮件。POP3 服务器接收的是用户发送至 SMTP 服务器的邮件。

以上协议只是 Web 上一些常用的通信协议，除了这些协议以外，还有 RIP 协议、NFS 协议和 DNS 协议等。

7.3 Web 客户端技术

Web 客户端的主要任务是通过浏览器来展现页面内容并实现与用户的交互，HTML 是信息展现的有效载体。为了提高客户端的功能性，出现了脚本、DOM 等概念；为了提高开发人员对信息展现格式的控制能力，出现了 CSS 等技术。常用的客户端开发技术主要有 HTML、CSS 等基本开发技术，客户端脚本、DHTML、DOM 等主要开发技术和 ActiveX、

Applet 等扩充开发技术，其开发的代码在客户端(主要是浏览器)中执行。Web 客户端开发的主要任务是设计与开发 Web 应用的 Web 页面，通过设计格式、布局以实现数据的展现和用户交互。以下将简单分析 HTML、CSS 和 DOM 等一些客户端常用开发技术。

1. HTML/HTML5

HTML(HyperText Markup Language，超文本标记语言)是一种通用的标记语言，标记用符号“<”和“>”括起来。它简单易懂、易于实现，允许 Web 页面设计人员建立文本与图片相结合的复杂页面。这些 Web 页面可以被网上任何人浏览到，而不管使用的是什么类型的计算机或浏览器。HTML 定义了很多标记来表示不同的语义，如<H1>表示一级标题。

HTML 也有以下几个主要缺点。

(1) 表现过于简单。HTML 文件将数据和数据的展示集中在一起，形式较为简单，尽管具有表达脚本、表格等功能，但很难表现复杂的形式。

(2) 链路容易中断。链宿地址改变后，链源不能自动纠正。

(3) 检索时所花的时间较长。检索到的内容针对性较差，返回的结果较多。

(4) 扩展性差。HTML 的标记集合是固定的，不允许用户自行定义他们自己的标识。由于网络技术发展得非常快，不断有新的数据格式的文档产生，这就要求要有一种比较灵活的标签机制才能满足网络信息不断发展的要求，但 HTML 不允许用户根据需要来创建新的标记，更无法表示许多特殊行业的数据。

(5) 缺乏语义性。HTML 是一种标记技术，不能很好地揭示信息内容的本质，计算机无法知道各段文本的确切含义。HTML 在设计上是用来展示内容和手工浏览 Web 页面的，不适合用作网络信息资源的自动化组织管理。

这些缺点通过一系列结合扩展来进行解决，如采用 CSS、XHTML、XML 等。

随着 Web 标准和技术的发展，出现了 HTML 的新一代版本 HTML5。HTML5 的目的不是为了内容展示，而是为了支撑广泛的 Web 应用，因此，它支持新的元素、结构和语义。HTML 5 是近十年来 Web 开发标准最巨大的飞跃，它并非仅仅用来表示 Web 内容，它的新使命是将 Web 带入一个成熟的应用平台。在 HTML5 平台上，视频、音频、图像、动画，以及与计算机的交互都被标准化。HTML5 功能强大，让内容创作者只要使用基于标准的 HTML5，就能使页面使用可扩展图形、动画和多媒体。

HTML5 主要是基于 HTML、CSS、DOM 以及 JavaScript，减少对外部插件(如 Flash)的需求，改进错误处理机制，添加更多的取代脚本的标记，独立于设备。新的特性如用户绘画的 canvas 元素，用于媒体回放的 video 和 audio 元素，对本地离线存储有更好的支持，article、footer、header、nav 和 section 等新的特殊内容元素，calendar、date、time、email、url、search 等新的表单控件，<form>和<input>元素的新属性，等等。例如，在 HTML5 中显示视频，只需要写如下代码：

```
<video src = "news.ogg" controls = "controls"> 浏览器不支持 video 标签
</video>
```

controls 属性表示由客户端浏览器添加播放、暂停和音量控件。该标签还可以包含其他一些属性，如宽度和高度属性等，用于指定播放窗口的大小。

2. XHTML

XHTML(The Extensible HyperText Markup Language,可扩展超文本标识语言)是W3C的标准之一,最早叫做"HTML in XML",是作为一种XML应用被重新定义的超文本标记语言,是一系列当前和将来的文档类型和程序块,由HTML 4再生和扩展而来,但却是更严格更纯净的HTML版本。

XHTML系列文档基于XML,最终被设计用来与基于XML的用户代理程序一起工作。由于引入了命名空间(Name Space),故XHTML具有良好的可扩展性,可以根据用户的需要和浏览器的处理能力选用合适的DTD,用户可以在XHTML文档中任意添加自己需要的标记。XHTML可重用性强,XHTML的推出原本是作为XML的过渡标准,但从根本上讲,XHTML是一种过渡技术,结合了部分XML的强大功能及大多数HTML的简单特性。

XHTML和CSS结合可以使实现样式与内容相分离,同时还能很好地结合Web页面代码。在其他单独的文件中,还可以混合各种XML应用,比如MathML和SVG等。

3. DHTML

DHTML(Dynamic HTML,动态HTML)是近年来Web发展进程中最具实用性的创新之一,它是无标准的,是一种通过各种技术的综合发展而得以实现的概念,这些技术包括JavaScript、VBScript、DOM、Layers和CSS等。

DHTML就是当Web页面从Web服务器下载后无须再经过服务器的处理,而在浏览器中直接动态地更新Web页面的内容、排版样式、动画等。比如,当鼠标移至文章段落中,段落能够变成蓝色,或者当用户点击一个超链后会自动生成一个下拉式的子超链目录。

DHTML建立在原有技术的基础上,可分为如下三个方面。

(1) HTML(XHTML)。也就是Web页面中的各种页面元素对象,它们是被动态操纵的内容。

(2) CSS。CSS属性也是动态操纵的内容,从而获得动态的格式效果。

(3) 客户端脚本(如JavaScript)。它实际操纵Web页上的HTML和CSS。

也就是说,DHTML是HTML、浏览器对象模型结构、CSS和Script的综合。DHTML的特点可以总结如下。

(1) 动态内容。通过浏览器与Web页面文字的对象模型,Web页面不用下载,其内容与对象 可以动态地增加、删除和改变显示内容。

(2) 动态样式。CSS除了可以扩展HTML标记的样式属性外,还可以通过脚本程序来改变这些属性。传统Web页面的内容和样式编排,在下载到浏览器后,是固定的。相比之下,DHTHL通过脚本可以改变其字体、颜色,甚至是样式的编排内容。

(3) 实时定位。DHTML可以动态改变元素CSS样式中定位有关的属性,使对象、图片和文本在Web页面中移动,达到动画效果。

DHTML同样也会带来安全问题,如用户界面伪装。对于用户而言,他们看到的不一定就是真实的。使用DHTML技巧完全可以达到蒙骗用户的目的,比如流行的Clickjacking技术(也叫UI Redress,属于用户界面伪装技术的一种)。

4. CSS

CSS(Cascading Style Sheets,层叠样式表)是由W3C定义和维护的标准,是一种用来为结构化文档(如HTML文档或XML应用)添加样式(字体、间距和颜色等)的计算机语言。

一个Web页面的读者和作者都可以使用CSS来决定文件的颜色、字体、排版等显示特性。CSS最主要的目的是将文件的结构(用HTML或其他相关的语言写的)与文件的显示(CSS)分隔开来。CSS还可以使用其他的显示方式,比如声音(如浏览器有阅读功能)或给盲人用的感受装置。此外CSS还可以与XHTML、XML或其他结构文件一起使用,唯一条件是显示这种文件的浏览器能够接受CSS的功能。采用CSS布局相对于传统的Web页面布局具有以下4个显著优势。

(1) 表现和内容相分离。将设计部分剥离出来放在一个独立样式文件中,HTML文件中只存放文本信息,这样的Web页面对搜索引擎更加友好。这个分隔有如下好处。

① 文件的可读性加强。

② 文件的结构更加灵活。

③ 作者和读者可以自己决定文件的显示。

④ 简化了文件的结构。

(2) 提高Web页面浏览速度。对于同一个Web页面视觉效果,采用CSS布局的Web页面容量要比用表格(<TABLE>)编码的Web页面文件容量小得多,前者一般只有后者的1/2大小。这样浏览器就不用去编译大量冗长的标签。

(3) 易于维护和改版。CSS与Web页面分离的特性,使得对于Web页面,只要简单地修改几个CSS文件就可以重新设计整个Web应用的页面,使得Web页面更加容易维护和改版。比如,新闻系统为了迎接春节要将背景颜色改得喜庆,只需修改CSS中的背景颜色为红色(或类似表达颜色)或庆祝图片,即可达到所有页面的背景色都变得迎合节日气氛的效果。如某大学为了迎接校庆而修改的背景图片的CSS为:

```
background: url("../images/2011xq/bg.jpg") repeat scroll 0 0 transparent;
```

(4) 使用CSS布局更符合现在的W3C标准。由于CSS标准自身是由W3C制定的,所以使用CSS布局完全满足W3C的标准。而且,随着近些年浏览器的推出,CSS规范得到了进一步的发展,目前广泛使用的是CSS3。

5. Flash/Flex 技术

1) Flash

Flash是Macromedia公司(现已被Adobe收购)出品的动画制作程序,利用Flash可以制作出一种后缀名为swf的文件,其中包含了声音、图像和动画等。这种文件既可加入HTML中,又可单独作为页面使用。

Flash的优点是其生成的多媒体文件是矢量图,体积小,还可以边下载边播放,这样避免了用户长时间等待。而且,用Flash生成的文件是代码保护的,别人无法看到其源代码,还可以禁止下载。

Flash使用ActionScript编程语言进行编写。ActionScript是Macromedia为Flash产

品开发的，最初是一种简单的脚本语言，目前最新版本3.0是一种完全的面向对象编程语言，其语法和JavaScript很相似，适用于Flash平台的各种应用的开发，从简单的动画，到复杂的、富数据的、交互性的各种网页和RIA应用程序。

2）Flex

Flex是Macromedia发布的展示服务的新版本，根据.mxml文件（纯粹的XML描述文件和Action Script）产生.swf文件，由flash player或者shockwave player解释执行，以提供丰富的客户体验，是重要的RIA技术。

Flex通过Java或者.NET等非Flash途径解释.mxml文件组织组件（component），并生成相应的.swf文件。Flex的component和Flash的component很相似，但有所改进和增强。

运用Flash完全可以做到Flex的效果。但是Flash天生是为了designer（设计者）设计的，界面和Flash的动画概念和编写程序的开发人员格格不入。Flex是为了吸引更多的熟悉JSP、ASP和PHP等编程的developer（开发者）而推出的，用更加规范和标准化的.mxml描述界面。为了避免理解混乱，从4.0版本开始，Flex Builder更名为Flash Builder，而Flex SDK名称仍然不变，以明确Flash Builder是面向设计者的设计工具，而Flex SDK是面向开发者的开发语言或者说开发框架。

6. DOM

DOM（Document Object Model，文件对象模型）是W3C推荐的处理可扩展标记语言的标准编程接口，主要作用是建立Web页面与Script或程序语言沟通的桥梁。自从W3C建立了DOM标准（W3C DOM）以及DOM和浏览器兼容之后，DOM在实际应用中使用得越来越广泛。

DOM常用来和JavaScript交互，即程序以JavaScript编写，但使用DOM来存取页面及其元素。这两者之间的结合非常紧密，甚至可以说，如果没有DOM，人们在使用JavaScript的时候是不可想象的，因为每解析一个结点一个元素都要耗费很多精力。DOM本身是设计为一种独立的程序语言，以一致的API存取文件的结构表述。

在和JavaScript进行交互的时候，DOM主要用来解析XML文档。当应用程序需要不断地导航、修改文档或随机地一次访问整个文档时，一般就使用DOM来解析。在使用DOM进行解析的时候，它在内存中构建起一棵完整的解析树，借此实现对整个XML文档的全面、动态访问。也就是说，它的解析是有层次的，即将所有的HTML中的元素都解析成树上层次分明的结点，然后可以对这些结点进行增加、删除、修改和查找等操作。

DOM的使用非常简单。用户可以随机地访问XML文档，由于整个树都构建在内存中，因此可以通过DOM应用程序接口修改这些结点，例如增加一个子结点或修改、删除一个结点。

DOM的优势主要表现在易用性强。使用DOM时，将把所有的XML文档信息都存于内存中，并且遍历简单，支持XPath（XPath是一种在XML文档中查找信息的语言，XPath用于在XML文档中通过元素和属性进行导航），增强了易用性。

虽然内存树结构提供了很好的导航支持，但仍有一些解析策略问题需要仔细考虑。首先，整个XML文档必须一次解析完成，不可能只做部分解析，因而，效率低，解析速度慢，内

存占用量过高，对于大文件来说几乎不可能使用；其次，效率低还表现在大量地消耗时间。因为使用DOM进行解析时，将为文档的每个元素、属性、处理指令和注释都创建为对象，这样在DOM机制中所运用的大量对象的创建和销毁无疑会影响其效率；第三，一般的DOM结点类型在互操作性上有优势，但对于对象类型绑定也许不是最好的。

7. JavaScript/AJAX

1) JavaScript

JavaScript是一种广泛应用于Web客户端开发的脚本语言，其源代码以文本格式的字符代码发送给客户端浏览器，发往客户端后由浏览器解释运行。JavaScript是一种动态、弱类型、基于原型的语言，内置支持类型，其语法类似Java，一些名称和命名规范也借鉴Java。常用于为HTML页面添加动态功能，操纵Web页面上的元素，以实现Web页面的客户端交互功能，制作特殊动态效果。例如，注册时用户输入信息的有效性验证、弹出的信息框、鼠标指针的文字跟随、渐隐渐现的图片等丰富Web页面表现的各种操作。因此，在Web页面设计中得到了广泛的应用。

但是解释语言的弱点是安全性较差。而且，在JavaScript中，如果一条语句不执行，那么后面的语句也无法执行。另外由于每次重新载入都会重新解释，载入后有些代码还会延迟至运行时才解释，甚至多次解释，所以，一定程度上速度较慢。

2) AJAX

AJAX(Asynchronous JavaScript and XML，异步JavaScript和XML)是一种创建交互式Web应用的网页开发技术，它有机地利用了一系列相关技术。它的一些部分以前称为动态HTML(Dynamic HTML)和远程脚本(Remote Scripting)。技术上，AJAX极大地发掘了Web浏览器的潜力，开启了大量新的可能性，目前广泛应用于RIA应用开发。典型的例子是，Google在它著名的交互应用程序中使用了异步通信，如Google讨论组、Google地图、Google搜索建议、Gmail等。AJAX的使用满足如下几点。

① 基于Web标准XHTML结合CSS来表示信息。

② 使用JavaScript操纵DOM进行动态显示及交互。

③ 使用XML和XSLT进行数据交换及相关操作。

④ 使用XMLHttpRequest进行异步数据查询、检索。

⑤ 使用JavaScript将所有的东西绑定在一起。

AJAX的应用使用支持以上技术的Web浏览器作为运行平台，如Mozilla、Firefox、IE、Opera、Konqueror及Safari。

AJAX Web应用模型使客户端在执行屏幕更新时，为用户提供了很大的灵活性，其优势主要表现如下。

① 减轻服务器的负担。AJAX的原则是"按需取数据"，可以很大程度地减少冗余请求和响应对服务器端造成的负担。

② 无刷新更新页面，减少用户心理和实际的等待时间。使用AJAX能在不更新整个页面的前提下维护数据，使得Web应用程序更为迅捷地回应用户动作，并避免了在网络上发送那些没有改变过的信息，当要读取大量数据时特别有用。AJAX使用XMLHTTP对象发送请求并得到服务器响应，在不重新载入整个页面的情况下用JavaScript操作DOM最

终更新页面。所以在读取数据的过程中，用户所面对的不是白屏，是原来的页面内容（也可以加一个加载的提示框让用户知道处于读取数据过程），只有当数据接收完毕之后才更新相应部分的内容。

③ 带来更好的用户体验。使用AJAX使用户不用面对白屏，等待时间减少，提高了用户体验。

④ 减轻了服务器端压力。AJAX可以把以前一些服务器负担的工作转嫁到客户端，利用客户端闲置的能力来处理，减轻服务器和带宽负担。

⑤ 可以调用外部数据。

⑥ 不需要下载插件或小程序。基于标准化的并被广泛支持的技术，不需要下载插件或小程序。

⑦ 进一步促进页面呈现和数据的分离。AJAX在整个Web服务系统的位置决定了AJAX引擎只要从服务端获取XML或者其他格式的数据，便可定制整个Web界面。从而可以使服务端注重数据逻辑处理而不必关心Web界面的呈现，将数据呈现的工作交给AJAX引擎来完成，这样有利于分工合作，减少非技术人员对Web页面的修改造成的Web应用错误，提高效率，更加适用于现在的分布式系统。

8. ActiveX/Silverlight

1）ActiveX

根据微软软件开发指南MSDN（Microsoft Developer Network，微软面向软件开发人员的一种信息服务）的定义，ActiveX插件以前也称为OLE控件或OCX控件，是一些软件组件或对象，如多媒体效果、交互式对象以及复杂程序等内容，可以将其插入到Web页面或其他应用程序中。它是一个开放的集成平台，为开发人员、用户和Web生产商提供了一个快速而简便的在Internet和Intranet创建程序集成和内容的方法。

ActiveX在广义上是指微软的整个COM架构，但是现在通常用来称呼基于标准COM接口来实现对象连接与嵌入的ActiveX控件。后者是指从VBX发展而来的，面向微软的IE技术而设计的以OCX为扩展名的OLE控件。通过定义容器和组件之间的接口规范，如果编写了一个遵循规范的控件，那么可以很方便地在多种容器中使用而不用修改控件的代码。同样，通过实现标准接口调用，一个遵循规范的容器可以很容易地嵌入任何遵循规范的控件。由于OLE在ActiveX控件中的应用的普及，现在OLE技术中只有少数独立于ActiveX技术，如复合文档。

IE等一些浏览器都在不同程度上支持ActiveX控件。这允许Web页面通过脚本和控件交互产生更加丰富的效果，同时也带来一些安全性的问题。IE和一些其他应用程序同时支持ActiveX文档接口规范，允许在一个应用程序中嵌入另一个支持这个规范的应用程序。很多应用软件，例如微软的Office系列和Adobe的Acrobat Reader都实现了这个规范。

2）Silverlight

Silverlight是微软用于Web前端应用程序开发的解决方案，其界面用XAML描述，后端可以用任何一种.NET兼容的语言开发，是跨浏览器、跨客户端平台的Web开发技术，能够设计、开发和发布有多媒体体验的RIA应用。

Silverlight以浏览器的外挂组件方式，提供Web应用程序中多媒体（含影音流与音效

流）与高度交互性前端应用程序的解决方案。通过 Silverlight 能够开发出具有专业图形、音频和视频的 Web 应用程序，增强了用户体验。Silverlight 集成了多种现有 Web 技术和设备，它可以在 Windows、Mac 平台上运行，支持 IE、Firefox，甚至 Apple 的 Safari 浏览器，而无须对现有的 Web 应用设计进行移植，甚至包括利用 Adobe Flash 设计的内容。

9. Applet/JavaFX

1）Applet

Applet 就是用 Java 语言编写的一些小的应用程序，它们可以直接嵌入到 Web 页面中，并能够产生特殊的效果。当用户访问这样的 Web 页面时，Applet 被下载到用户计算机上执行，但前提是用户使用的是支持 Java 的浏览器。由于 Applet 是在用户计算机上执行的，因此它的执行速度是不受网络带宽限制的，用户可以更好地欣赏 Web 页面上 Applet 产生的多媒体效果。

在 Applet 中，可以实现图形绘制、字体和颜色控制、动画和声音的插入、人机交互及网络交流等功能。Applet 还可以利用用户计算机的 GUI 元素，可以建立标准的图形用户界面，如窗口、按钮、滚动条等。

2）JavaFX

Applet 逐渐淡出市场，Sun 公司 2007 年发布了针对 RIA 的技术或者平台 JavaFX。有人称 JavaFX 为“下一代的 Applet”。JavaFX 包括 JavaFX 脚本语言和 JavaFX Mobile 应用。JavaFX 脚本语言是 Sun 开发的一种声明性（Declarative）、静态类型的脚本语言，其功能丰富，类 Java 语法可以使用户非常简单地快速开发出强大的、功能性更强的、更为安全的富互联网应用程序（RIA）。

JavaFX 具有结构化代码、重用性和封装性等特性，例如包、类、继承和单独编译与发布单元，它为多种多样的操作提供了声明式、无中间程序逻辑的语法，这些操作包括创建 2D 动画、设置属性或者声明在模式和视图对象之间的绑定依赖关系。

JavaFX 扩展了 Java 平台在图形、动画以及高保真音频和视频等方面的功能，大大缩短了 Java 开发人员和 Web 设计人员的开发周期，使创建集图形、视频、音频、动画和丰富内容于一体的应用变得非常简单。它为桌面、浏览器和移动设备上富于表现力的 RIA 的创建提供了一个统一的开发和部署模式。

10. VRML 与 X3D

1）VRML

VRML（Virtual Reality Modeling Language，虚拟现实建模语言）是目前 Internet 上基于 WWW 的三维互动站点制作的主流语言，用来描述三维物体及其行为，以构建虚拟境界。VRML 的基本目标是建立 Internet 上交互式的三维多媒体，具有分布式、三维、交互性、多媒体集成和境界逼真性等基本特征。

VRML 的对象称为结点，结点的集合可以构成复杂的景物。结点可以通过实例得到复用，对它们赋以名字，进行定义后，即可建立动态的虚拟现实。

VRML 不仅支持数据和过程的三维表示，而且能提供带有音响效果的结点，用户能走进视听效果十分逼真的虚拟世界（如简易迷宫、国际象棋）。用户使用虚拟对象表达自己的

观点，能与虚拟对象交互，为用户对具体对象的细节、整体结构和相互关系的描述带来的新的感受。

2）X3D

X3D(Extensible 3D，可扩展三维语言)，是 VRML 标准的升级版。X3D 基于 XML 格式，可以直接使用 XML DOM、XML Schema 校验等技术和相关的 XML 编辑工具，包括了更强大、更高效的 3D 计算能力、渲染质量和传输速度。可以使用延迟着色技术，支持特效 SSAO 和 CSM 阴影、实时环境反射和折射、基于实时环境和天光的光照、HDR、运动模糊、景深。X3D 支持对应 3ds MAX 标准材质的多种贴图。

11. XML

XML(Extensible Markup Language，可扩展标记语言)是 W3C 制定的一种简单的、跨平台的、依赖于内容的技术，是目前处理结构化文档信息的有力工具。XML 文档可以是服务器端的文件，也可以是如 Web 服务环境中的计算机系统之间传输的数据。XML 的可扩展性强，是一种元语言，可用于创建或定义其他个性化的标记语言，例如 RSS、MathML 甚至 XSLT。XML 的标记可以自定义，提供更多的数据操作。

XML 与 HTML 类似，其标记用"<"和">"括起来。XML 与 HTML 的设计区别在于 XML 是用于数据存储，重在数据本身的描述；XML 的语法比 HTML 更加严格。XML 的简单使不同应用程序可以灵活地交换信息。

XML 最大的特点是其文档类型和可移植性。XML 文档具有不同类型，XML 的组成部分及 XML 的结构定义了文档的类型。XML 文档都采用相同的字符编码模式，使其可以在不同的计算机环境中移植。

XML 给 Web 应用赋予了强大的功能和灵活性，主要表现如下。

① 更有意义的搜索。数据可被 XML 唯一标识。例如，有了 XML，图书就可以很容易以标准的方式按照作者、标题、ISBN 序号或其他的标准分类，搜索图书就变得十分方便。

② 开发灵活的 Web 应用。由 XML 表示的数据可以发送到其他应用软件、对象或者中间层服务器做进一步的处理，或者可以发送到桌面用浏览器浏览。XML 和 HTML、脚本、公共对象模式一起为灵活的多层 Web 应用软件的开发提供了所需的技术。

③ 不同数据源的信息集成。现在流行的数据库产品众多，XML 使不同数据源的结构化的数据集成简化。如在包装器-中介器信息集成模式中，包装器将数据源的数据进行包装，表示为 XML 格式，中介器进行转换和模式匹配，然后发送到客户或其他服务器做进一步的处理和分发。

④ 数据的多样显示。由于数据显示与内容分开，XML 定义的数据允许指定不同的显示方式，使数据更合理地表现出来。本地数据能够以客户配置、使用者选择或其他标准决定的方式动态地表现出来。CSS 和 XSL 为数据的显示提供了开放的机制。

⑤ 在 Web 上发布数据。由于 XML 是一个开放的基于文本的格式，它可以和 HTML 一样使用 HTTP 进行传送，不需要对现有的网络进行变化。

⑥ 压缩性。因为 XML 用于描述数据结构的标签可以重复使用，所以其压缩性能好。当服务器与客户间数据的传递量大，而且应用程序在运行压缩时，即可对 XML 数据进行压缩。

⑦ 开放的标准。XML 基于的标准是为 Web 进行过优化的，而且一些公司以及 W3C 工作组正致力于确保 XML 的互用性，以及为开发人员、处理人员和不同系统与浏览器的使用者提供支持，并进一步发展 XML 的标准。

⑧ XML 使用前景广泛。主要表现在：商业智能；丰富的标签完全可以描述不同类型的单据，可以被加密、附加上数字签名等；信息发布；客户可按需选择和制作不同的应用程序以处理数据，构成广泛的、通用的分布式计算系统；智能化的 Web 应用和信息集成；XML 能更准确地表达信息的真实内容，严格的语法降低了应用程序的负担，也使智能工具的开发更为便捷。

Web 本身就是一个最大的分布式应用系统，因而，XML 技术在 Web 上的优势明显。Web 应用借助 XML 格式交换信息，能很好地解决分布式架构上的信息交换。XML 在 Web 应用服务器端应用很广。



7.4 Web 服务器端技术

7.4.1 Web 应用服务器端开发技术

Web 服务器端的开发技术也是由静态向动态逐步发展完善起来的。Web 服务器端开发的代码在服务器(Web 服务器及扩展环境)端执行，主要有 CGI、ISAPI/NSAPI 等基本开发技术，PHP、ASP、ASP. NET、JSP 等高级开发技术和基于后台数据库的 ODBC、ADO、JDBC 等数据库连接技术。每种技术都有其特点，开发人员要根据需求和具体情况来决定使用哪种系统环境和开发技术。这些技术的主要任务是设计与开发应用程序，在处理用户交互的基础上实现既定的需求，为客户端提供动态生成的数据，实现数据源之间的交互与管理。

1. CGI

CGI(Common Gateway Interface，公共网关接口)是一种早期应用于 Web 应用的标准，用于 Web 服务器运行服务器端外部程序，称为 CGI 脚本，多用于动态生成 Web 内容。遵循这套接口标准，可以使用任何编程语言来编制 CGI 程序，而不受开发语言的限制，例如 C、C++、VB、Delphi 等。与静态的 Web 获取不同，使用 CGI 可以创建程序，当用户发出请求时，Web 服务器将请求转到目标程序，执行程序后接收其输出，并将输出作为 HTTP 响应发回给客户端。CGI 程序是独立的可执行程序，每次收到 CGI 请求时由 Web 服务器调用，服务器进程与 CGI 进程进行进程间通信。用于服务器端 CGI 编程的流行语言有 Perl。

但 CGI 程序的开发效率和运行效率都不高，开发的难度较大，使用不方便，而运行时由于需要单独的进程，所以开销较大。

2. ISAPI

ISAPI(Internet Server Application Programming Interface，因特网服务器应用程序接口)是微软为克服 CGI 的低效性而开发的运行在 Windows 和 IIS 上的一组 API，ISAPI 服务器扩展是可以被 HTTP 服务器加载和调用的动态链接库。

ISAPI服务器扩展使用户可以填写窗体，然后单击提交按钮将数据发送到Web服务器并调用ISA，ISA处理这些信息以提供自定义内容或将这些信息存储在数据库中。Web服务器扩展可以使用数据库中的信息动态生成Web页面，然后将其发送到客户计算机进行显示。应用程序可以使用HTTP和HTML添加其他自定义功能并将数据提供给客户端。服务器扩展和筛选器均在Web服务器的进程空间中运行，这样就为扩展服务器的功能提供了有效的手段。

但是ISAPI编程复杂，开发效率低，而且，由于ISAPI和Web服务器运行在同一个进程内，如果一个ISAPI程序出现了致命的错误，有可能造成整个Web服务器瘫痪。

3. PHP

PHP(Hypertext Preprocessor，超级文本预处理语言)是一种HTML内嵌式脚本语言，在服务器端执行。PHP的语法混合了C、Java、Perl以及PHP独创的语法。它可以比CGI或者Perl更快速地执行动态Web页面。用PHP制作出的动态页面与其他的编程语言相比，执行效率比完全生成HTML标记的CGI要高许多，PHP执行引擎还会将用户经常访问的PHP程序驻留在内存中，其他用户再一次访问这个程序时就不需要重新编译程序，直接执行内存中的代码即可。PHP功能强大，几乎所有的CGI功能PHP都能实现，而且支持几乎所有流行的数据库以及操作系统。PHP的优势主要表现在以下几个方面。

① 学习快速。PHP是一种能快速学习、跨平台、有良好数据库交互能力的开发语言，让UNIX/Linux有了一种可以与ASP媲美的资本。

② 与Apache及其他扩展库紧密结合。PHP可以与Apache以静态编译的方式结合起来，而与其他的扩展库也可以用这样的方式结合(Windows平台除外)。这种方式极大地利用了CPU和内存，有效地利用了Apache的高吞吐能力。同时，外部的扩展也是静态联编，从而达到最快的运行速度。由于与数据库的接口也使用了这样的方式，所以使用本地化调用，这也让数据库发挥了最佳效果。

③ 良好的安全性。因为PHP本身的代码开放，所以其代码在许多工程师手中进行检测，具有了公认的安全性能。此外，PHP为应用开发也提供了一些内置的安全性支持，典型的如mysql_real_escape_string函数。

目前有很多基于UNIX(或Linux)服务器的Web应用是使用PHP技术开发而成。

4. ASP/ASP.NET

ASP和ASP.NET是基于Windows开发Web应用的重要并被广泛使用的技术。

1) ASP

ASP(Active Server Page，动态服务器页面)是微软开发的代替CGI脚本程序的一种，是嵌入式网页中的脚本，可由服务器执行的服务器端脚本技术。ASP可以与数据库和其他程序进行交互，是一种简单、方便的编程技术框架。微软把ASP描述为："一个服务器的脚本环境，在这里可以生成和运行动态的、交互的、高性能的Web服务器应用程序"。ASP的主要特性是能够把HTML、脚本和组件等有机地组合在一起，形成一个能够在服务器上运行的应用程序，并把按用户要求专门制作的标准HTML页面送给客户端浏览器。

ASP的主要特性表现如下。

① 利用 ASP 可以实现突破静态页面的一些功能限制，实现动态页面技术。

② ASP 文件包含在 HTML 代码所组成的文件中，易于修改和测试。

③ 服务器端的 ASP 解释程序会在服务器端执行 ASP 程序，并将结果以 HTML 格式传送到客户端浏览器上。因此使用各种浏览器都可以正常浏览 ASP 所产生的页面。

④ ASP 提供了一些内置对象，使用这些对象可以使服务器端脚本功能更强。例如可以从 Web 浏览器中获取用户通过 HTML 表单提交的信息，并在脚本中对这些信息进行处理，然后向 Web 浏览器发送信息。

⑤ ASP 可以使用服务端 ActiveX 组件来执行各种各样的任务，例如访问数据库和文件系统等。

⑥ 由于服务器是将 ASP 程序执行的结果以 HTML 格式传回客户端浏览器，因此使用者不会看到 ASP 所编写的原始代码，可防止 ASP 程序被窃取。

2）ASP. NET

ASP. NET 是用于动态 Web 应用构建的免费 Web 框架，包括很多开发 Web 应用所需的服务，作为. NET Framework 的一部分提供，可以使用与公用语言运行时（CLR）兼容的任何语言编写应用程序。使用这些语言可以利用 CLR、类型安全和集成等优点。编写的代码在服务器端首次运行时进行编译，在服务器上运行。ASP. NET 作为 Web 服务器的一部分，处理浏览器发出的请求，处理已经映射到其上的文件扩展名，如. aspx、. ascx、. ashx 和. asmx。使用 ASP. NET，一般以 IIS 作为服务器以提供完整的 Web 服务器功能。

通用语言的基本库、消息机制和数据接口的处理都能无缝地整合到 ASP. NET 的 Web 应用中。程序员可以选择一种或多种最适合的只要是兼容. NET 公共语言运行时所支持的语言来编写程序，包括 C＃、Visual Basic、JScript. NET、managed C++、J＃，其中 C＃在近些年更为常用。将来，多种程序语言协同工作的能力保护程序员现在的基于 COM＋开发的程序，能够完整地移植向 ASP. NET。

5. Servlet

Servlet 是一种由 Java 编写的服务器端程序，可以动态生成 Web 页面，是 Java EE 的一部分。Servlet 运行于支持 Java 的 Servlet 容器中，如 Java EE 的 Servlet 容器，在被调用后会被动态地载入到容器由容器解释执行。Servlet 的. java 文件先编译成. class 文件，然后部署在 Servlet 容器。

Servlet 支持 Request/Response 模型，在服务器端接收 Web 客户端的请求并给出响应，一般通过 HTTP 协议来完成。Servlet 容器提供了侦听请求的服务，并能把客户请求和响应信息包装在特殊的 Request/Response 对象中，交给 Servlet 处理，然后由 Servlet 处理请求并通过 HTTP 协议将响应通过容器转发到客户端。

在基于 MVC 模式的 Web 应用中，Servlet 充当控制器的角色，用来处理 HTTP 请求，管理应用的工作流程。Servlet 通过用户所发送的 HTTP 请求，接收用户的输入事件，并把这些信号翻译成消息传递给封装了请求业务逻辑的 JavaBean 或 EJB 进行交互，最后激活 JSP。

Servlet 的主要优势如下。

（1）Java 语言的优点。由于使用 Java 语言开发，具有 Java 语言的所有特性，如可移植

性、内存自动回收、良好的面向对象特性、异常处理机制、包括联网支持在内的大量应用程序接口和易用性等内容。

(2) 执行效率高。一个 Servlet 可以同时处理多个请求，每个请求将生成一个新的线程，多个客户能够在同一个进程中同时得到服务。

(3) 构造的控制器功能强。Servlet 是模块化的，每个模块执行一个特定的任务，也可以协同工作，构成功能更强的“Servlet 链”。

6. JSP

JSP(Java Server Page，Java 服务器页面)是一种将 Java 代码嵌入到 HTML 页面中实现的服务器端 Web 开发技术，在服务器端进行解析，动态生成页面传递给客户端，使 Web 应用开发人员可以使用 HTML 和 XML 模板以及 Java 代码在服务器上建立动态内容。同时，它还是一种安全、快速并且与服务器平台无关的设计方法。JSP 是 Java EE 的一部分，本质上是一种高层的 Servlet，从使用的角度而言，与其他页面编写脚本有很大的相似性。

在 JSP 中，HTML 代码主要负责描述信息的显示，而“＜％”和“％＞”之间的 Java 程序代码则用来描述处理逻辑。JSP 页面在服务器端在初次请求时转换成 Servlet，然后编译成字节码进行执行，并将执行结果重新嵌入到 HTML 代码中后发送给浏览器。客户端浏览器不需要任何附加的软件支持。

JSP 作为 Java EE 的部分，可以和 JavaBean、Servlet 或 EJB 结合使用，JSP 优点突出，可以使：①内容的生成和显示分离；②生成可重用的组件；③采用标签简化页面开发，通过开发定制标签库，实现了 JSP 技术的扩展；④具有 Java 的优点，包括健壮的存储管理和安全性，可靠且移植方便等；⑤企业产品多样性，在 Java EE 平台内容不仅包括管理复杂的企业应用程序而且包括事务管理技术和 pooling 资源管理技术。

7. Perl/Python/Ruby

1) Perl

Perl 是一种自由且功能强大的脚本语言，最初的设计者为 Larry Wall。Perl 借取了 C、sed、awk、Shell Scripting 以及很多其他编程语言的特性。其中最重要的特性是它内部集成了正则表达式和巨大的第三方代码库 CPAN。从最初被当作一种在跨平台环境中书写可移植工具的高级语言开始，Perl 就已经被广泛地认为是一种工业级的强大工具，可以在任何地方用来完成工作。

Perl 逐步发展成为一种功能齐全的程序设计语言，特别是在各种计算平台上，它被用作 Web 编程、数据库处理、XML 处理以及系统管理。它能够完成所有这些工作，同时仍然是处理小的日常工作的完美工具，这是它的设计初衷。Perl 快速而且特别有用。随着 Perl 的日益流行，Perl 实际上已经被所有 UNIX(包括 Linux)捆绑在一起作为标准部件发布，而且也被广泛用于 Windows 和几乎所有其他的操作系统。Amiga、BeOS、VMS、MVS 和 Apple Macintosh 等也只是 Perl 已经完成移植的平台的一小部分。

2) Ruby

Ruby 是 20 世纪 90 年代开发出的一种面向对象、解释型脚本语言，综合了 Perl、Python 和 Java 等语言的特点，语法简单，擅长文本处理和系统管理等任务，还有异常处理以及迭代

器等构造，这些特性使编程简单明了。Ruby on Rails (RoR)Web 应用框架出现后，Ruby 即被广泛地应用于 Web 应用的开发中。

3) Python

Python 是一种面向对象、解释型计算机程序设计语言，也是一种功能强大而完善的通用型语言。它具有非常简捷而清晰的语法特点，适合完成各种高层任务，几乎可以在所有的操作系统中运行。Python 支持最新的 XML 技术，可用于 Web 应用的开发。

Python 具有很多优秀的脚本语言的特点：解释的、面向对象的、内建的高级数据结构，支持模块和包，支持多种平台、可扩展，还支持交互式或图形方式运行，拥有众多的编程界面，支持各种操作系统平台以及众多的各类函数库。与其他脚本语言相比，Python 程序员可以借助 Python 语言提供的 API，使用 C 或者 C++来对 Python 进行功能性扩展，从而既可以利用 Python 方便灵活的语法和功能，又可以获得与 C 或者 C++几乎相同的执行性能，同时 Python 通过与 C 语言的有机结合有效解决了脚本语言执行慢问题，从而使脚本语言的应用范围得到了很大扩展。

Python 在服务器端的编程可以通过 Web 应用服务器端开发框架加以支持，也可以通过 CGI 脚本编程，另外，也有使用 Python 的服务器和内容管理系统的实现。典型应用如 Google 的搜索引擎大量使用 Python，Zope 应用服务器和著名的 BT 下载工具 Bit Torrent 由 Python 编写。

事实上，Python 也可以用于客户端。Web 浏览器中的插件提供对 DOM 模型的支持，也可以使用插件或转换成 JavaScript 等方式来支持 Python，然后在页面中使用 <script type="text/python"> 来替换 JavaScript 的使用方式 <script language="javascript"/>即可。

8. ColdFusion

ColdFusion 最早是由 Allaire 公司开发的一种应用服务器平台，其运行的 CFML 或 CFM(ColdFusion Markup Language，ColdFusion 标记语言)是针对 Web 应用的一种脚本语言，类似现在的 JSP 里的 JSTL(JSP Standard Tag Lib)，从 1995 年开始开发，其设计思想被一些人认为非常先进，被一些其他语言所借鉴。

ColdFusion 文件以 *.cfm 为文件名，在 ColdFusion 专用的应用服务器环境下运行。在 Allaire 公司被 Macromedia 公司收购以后，推出了 Macromedia ColdFusion 5.0。类似于其他的应用程序语言，cfm 文件被编译器翻译为对应的 C++语言程序，然后执行并向浏览器返回结果。

自 Macromedia 接收 Allaire 公司后，把原来基于 C++开发的 ColdFusion 改为基于 JRun 的 Java EE 平台的一个 Web 应用(JRun 也是 Allaire 公司的一个 Java EE 服务器产品)，并正式推出 Macromedia ColdFusion MX 6.0 版本。此时的 cfm 运行原理就和 Java 非常地类似，cfm 文件被应用服务器编译为对应的 .java 代码并编译成 .class 文件在 Java 虚拟机上运行。从此 ColdFusion 完全从一个功能齐全的动态 Web 服务器转变为一个 Java EE 应用服务器，同时依旧保留了原有版本的所有特性。

ColdFusion 的页面后缀通常为.cfm，同时 Macromeida 公司在发布 ColdFusion MX 的时候借鉴于 Java 面向对象设计风格，设置了.cfc 这样的 ColdFusion 文件后缀，它们被称做

ColdFusion Components(CFC 组件)。cfc 文件就好比一组 cfm 函数的集合,使对应的代码具有高度的可重用性。虽然.cfc 和 custom tag 具有类似的重用性,但 cfc 提供了更加灵活的调用方式,例如 Web Service 方式的调用支持。

CFM 并不等同于 ColdFusion。CFM 是一种标记语言,而 ColdFusion 是一种应用服务器环境。对于标准的语法结构的 cfm、cfc 文件,它们不仅仅可以运行在 Macromedia ColdFusion 服务器上,同样也可以直接运行于 BlueDragon 服务器环境。

7.4.2 中间件技术

中间件是介于应用程序和应用软件或系统软件之间的软件或服务程序,它使用系统软件所提供的基础服务(功能),衔接网络上应用系统的各个部分或不同的应用,能够达到资源共享、功能共享的目的。通过中间件,应用程序可以工作于多平台或操作系统环境。中间件也是构成基于 XML、SOAP、Web 服务和 SOA 等新的信息技术的必要成分。通常,中间件必须具有以下特点。

(1) 支持标准的协议和接口。

(2) 支持分布式计算,提供跨网络、硬件和操作系统的应用或服务的透明交互。

(3) 满足大量应用的需要。

(4) 能运行于多种硬件和操作系统平台。

作为操作系统和应用程序接口之间的支撑软件,中间件可以屏蔽硬件、软件、协议和算法的复杂性和差异,便于升级和扩充业务能力,从而缩短应用程序的开发周期,节约应用的开发成本、减少系统初期的建设成本,降低应用开发的失败率,保护已有的投资,简化应用集成,减少维护费用,提高应用的开发质量,保证技术进步的连续性并增强应用的生命力。另外,中间件还可将不同时期在不同操作系统上开发的应用软件进行集成,协调工作。

在具体实现上,中间件是用 API(应用程序接口)定义的软件层,具有强大的通信能力和良好的可扩展性的分布式软件管理框架。

1. 应用服务器

Web 应用服务器是用来组建 Web 应用,支持 Web 应用的软件系统,和 3(N)层架构关系密切,主要处理业务逻辑。应用服务器提供开发和运行基于组件的分布式应用,提供一系列服务,如事务、资源池、负载均衡、命名和路径范围等。目前 Web 应用服务器软件有多种,常用的有 IIS、Apache、WebLogic Server 和 WebSphere。

1) IIS

IIS(Internet Information Server,因特网信息服务器)是微软公司的产品,它借助于 Windows NT 2000 Server 2003 操作系统在 PC 界处绝对优势,也是当今使用最广泛的 Web 应用服务器之一。由于它具有与操作系统的亲和性,并继承了微软产品一贯的用户界面,IIS 利用与微软 Proxy Server、Certificate Server、Site Server、BackOffice 以及其他应用程序紧密结合之便,成为功能强大、使用方便的 Web 应用服务器。同时,IIS 具有很高的执行效率、出色的安全保密性、易于管理以及启动迅捷等特点。它既可用于集成现有的应用方式,也可用于实施 Web 应用。IIS 安装简单,操作方便,能够负担高容量站点,有不少大型的

商务站点建立在IIS之上。

2）Apache

Apache是世界使用排名第一的Web应用服务器软件，它可以运行在几乎所有广泛使用的计算机平台上。由于其跨平台和安全性被广泛使用，是最流行的Web服务器端软件之一。新闻系统开发采用了Java EE平台，因此选用的应用服务器需要支持Java EE，作为免费而开源的应用服务器，Apache系列的应用服务器是试验型Web应用最佳选择。

3）WebLogic Server

WebLogic Server是一个企业Java EE应用服务器，它支持在可靠、安全、高可用和可伸缩的环境中部署关键任务应用程序。

4）WebSphere

WebSphere是IBM的软件平台。它包含了编写、运行和监视全天候的工业强度的随需应变Web应用程序和跨平台、跨产品解决方案所需要的整个中间件基础设施，如服务器、服务和工具。WebSphere提供了可靠、灵活和健壮的软件。

2. EJB

Enterprise Java Beans(EJBs)是一种基于组件的架构，用Java语言开发与平台无关的分布式应用，运行于EJB容器中，是Java EE的一部分。EJB是一种封装业务逻辑的可重用组件。这些组件利用容器提供的服务来管理业务对象的生命周期和操作命名服务，提供事务处理、安全机制以及持久化机制。EJB包括以下3种类型。

(1) Session Bean(会话Bean)。会话Bean表示客户端和远程服务组件之间的交互，当客户端通过激活远程服务组件以请求业务服务时，会话Bean进行响应。会话Bean分为有状态和无状态两种。有状态(Stateful)会话Bean在方法调用时可以保持会话状态，在客户端请求时，为客户端分配一个实例，并为该客户端保持该组件的状态。例如，客户的网上购物车，客户在开始购物时，从数据库中查询客户的具体信息，往购物车里添加商品或从里面删除商品、下订单等时调用的其他方法也可以使用所查询的客户的具体信息。有状态会话Bean的状态在发生会话终止、系统崩溃或网络故障时消失。无状态(Stateless)会话Bean没有内部状态，不跟踪记录从一个方法调用传递到另一个方法调用的信息，不同调用之间相互独立。客户端请求无状态会话Bean实例时，可以从容器保持的实例池中接收一个实例。另外，它还可以共享，容器可以保持数量较少的实例为很多客户端提供服务。

(2) Entity Bean(实体Bean)。实体Bean是复杂的业务实体对象，表示数据库中的持久性业务对象，可能使用几个相关的Java对象，并通过主键实现唯一性，特定字段可以成为持久性字段，如新闻系统中新闻记者记录。实体Bean可以在多个客户端之间共享，如某个作者的稿件实体Bean可以由多个客户端用于更新稿件的审阅意见。实体Bean可分为Bean管理持久性(BMP)和容器管理持久性(CMP)。BMP需要Bean自己管理字段存放内容，也要编写相关程序代码，用于程序员想自行控制所有行为。CMP不需要程序员撰写太多代码，EJB的程序代码是EJB容器根据部署描述符在部署EJB的时候产生，用于快速开发。

(3) Message Driven Bean(MDB,消息驱动 Bean)。MDB 为实现异步通信提供了比 JMS 更加简单的方法。和会话 Bean 或实体 Bean 的方法调用方式不同,MDB 接收异步 JMS 消息,可以主动被触发,并做出相对应的响应。容器处理 JMS 消息对象和主题所需的大部分设置工作,把消息发送给相关的 MDB。MDB 允许 Java EE 应用程序发送异步消息,这些消息由接收方应用程序进行处理。

根据需要开发的应用程序的要求,分布式应用中可以包括大量各类 EJB,如采用 Session Bean 实现应用逻辑,采用 Entity Bean 可以表示数据。EJB 的事务或数据库控制等很多属性定义在配置文件(也称为部署描述符),在装载入 EJB 容器时,由 EJB 容器通过配置文件来管理 EJB 组件。

3. 消息系统

消息系统为分布式环境中系统之间提供基于消息的异步通信。这些分布式系统之间通过交换消息进行通信。消息源可以等待目标系统的响应,也可以不等待目标系统的响应。点和点之间的通信,消息系统也是通过请求/响应进行通信,而如果是发布/订阅(Publish/Subscribe)通信,订阅者注册特定主题(Topic)的消息服务,然后接收所有的发布者所发布的有关这一主题的消息。

Java 消息服务(JMS)提供了异步消息传输机制,MDB 接收 JMS 的消息。换句话说,EJB 可以作为 JMS 消息的目标,以提供 JMS 和 EJB 之间的异步消息传输。MDB 可以很容易地使用消息与 EJB 和 Java EE 组件进行交互。

7.4.3 Web 服务

Web 服务主要是基于 XML 和 HTTPS,其通信协议主要基于 SOAP,如图 7.1 描述了 Web 服务体系中各关键的技术,包括 SOAP、WSDL、UDDI、BPEL4WS 等。

Process	BPEL4WS
Universal Description, Discovery, and Integration	UDDI
Services Description	WSDL
Messaging	SOAP
Extensible Markup Language	XML
Transport Protocols	HTTP、SMTP 等

图 7.1 Web 服务协议栈

1. SOAP

SOAP(Simple Object Access Protocol,简单对象访问协议)是一种基于 XML 的轻量级的协议,用于在分布式环境中进行信息交互,主要用于 Web 服务中。SOAP 使用 Internet 应用层协议作为其传输协议,这些协议包括 HTTP、SMTP 等。

SOAP 具有跨硬件平台、操作系统、编程语言和网络硬件平台的高度互操作性,这些特性为 XML 在异构分布式环境中交换信息提供了一个简单而有效的机制。SOAP 包括以下 4 个部分的内容。

(1) SOAP 封装：定义了一个整体框架用来描述消息内容、消息的处理者和消息的性质(消息为可选的或者必需的)。

(2) SOAP 编码规则：定义了数据的一种序列化编码机制，通过这样一个机制来定义应用程序中需要使用的数据类型，并可用于交换由这些应用程序定义的数据类型所衍生的实例。

(3) SOAP RPC 表示：定义了一个远程过程的调用和响应协定。

(4) SOAP 绑定：定义了一个使用底层传输协议来完成结点间交换 SOAP 封装的约定。

SOAP 消息基本上是从发送端到接收端的单向传输，但常常结合起来执行类似于请求-应答的模式。所有的 SOAP 消息都是用 XML 编码，一条 SOAP 消息就是一个包含有一个必需的 SOAP 的封装包、一个可选的 SOAP 标头和一个必需的 SOAP 体块的 XML 文档。SOAP 消息结构如图 7.2 所示。

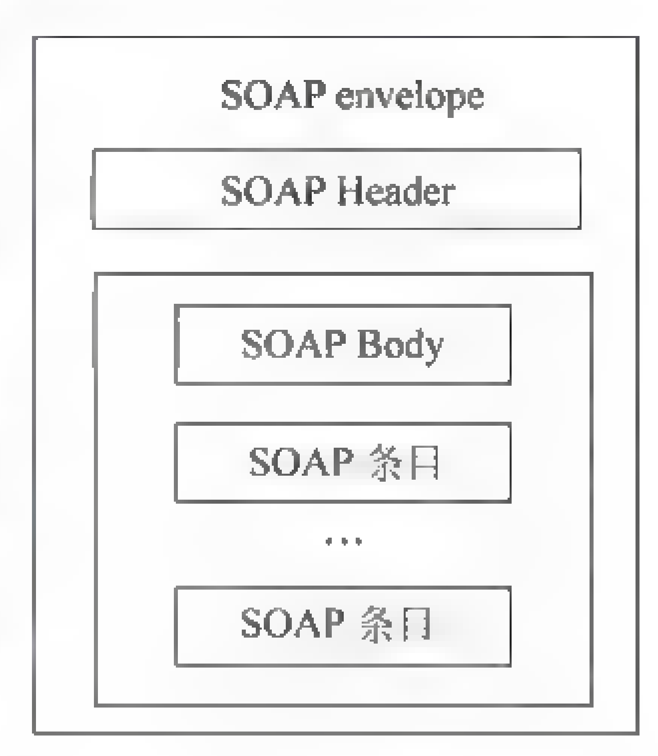


图 7.2 SOAP 消息结构

2. WSDL

WSDL(Web Services Description Language，Web 服务描述语言)是用 XML 文档描述 Web 服务的标准，是 Web 服务的接口定义语言，它由 Intel、IBM、Microsoft 等共同提出。它能够准确描述 Web 服务的功能和与 Web 服务进行通信的具体接口，并且用一种与具体语言无关的抽象方式定义了相应 Web 服务的具体操作、相关信息和 Web 服务的具体服务地址。

WSDL 服务定义为分布式系统提供了可由机器识别的 SDK 文档，并且可用于描述自动执行应用程序通信中所涉及的细节。从面向对象的角度来说，WSDL 描述了 Web 服务的功能函数、参数和返回值，通过 WSDL 可以明确以下几个基本属性。

① 服务的功能以及服务提供的相关操作。

② 服务交互的数据格式和交互的协议。

③ 服务的绑定信息。

④ 服务的具体地址。

为了帮助在注册中心发布和查找 WSDL 服务描述，WSDL 文档被分为服务接口和服务实现两部分，以使每个部分都可以独立定义并被其他部分重用。服务接口是 Web 服务的抽象定义，并被用于描述某种特定类型的服务，包括 Binding、Port Type、Message 和 Types 4 部分；服务实现文档包含实现一个服务接口的服务的描述，包括 Service 和 Port 两部分，一个服务实现文档可以包含对多个服务接口文档的引用。一个完整的 WSDL 服务描述由一个服务接口和一个服务实现文档组成。

表 7.1 显示了服务接口和服务实现所包含的元素。

表 7.1　服务接口和服务实现元素

分　类	名　称	作　用
抽象定义	消息(Message)	抽象定义了在通信中使用的消息的数据结构
	类型(Types)	数据类型的容器,包含了所有在消息定义中需要的 XML 元素的类型定义
	端口类型(Port Type)	定义了一种服务访问入口的类型,包含若干操作
	操作(Operation)	每个操作代表访问入口支持的一种类型调用,WSDL 支持包括单向请求、单向通知、请求响应、要求响应 4 种访问入口调用模式
具体定义	服务(Service)	描述一个具体的被部署的 Web 服务所提供的所有访问入口的部署细节,一个服务可包含多个服务访问入口
	绑定(Binding)	定义某个端口类型的具体协议和数据格式规范的绑定
	端口(Port)	描述服务访问入口的细节,包括地址、消息调用模式

3. UDDI

UDDI(Universal Description Discovery and Integration,统一描述、发现和集成服务)是一个全球化的、平台无关的开放式架构,用于描述、发现和集成相关的服务。它是一种目录服务,企业可以使用其对 Web 服务进行注册和查找,使得企业能够彼此发现,在 Internet 上进行交互。UDDI 通过使用一个全球性的注册中心,共享信息,加速全球 B2B 的电子商务的应用。

UDDI 以若干已建立的行业标准为基础,包括 HTTP、XML、XML Schema 定义、SOAP 和 WSDL,UDDI 规范描述了 Web 服务及其编程接口的注册表,它本身也是一组 Web 服务。

UDDI 的核心组件是 UDDI 注册,它使用一个 XML 文档来描述企业及其提供的 Web 服务。一般而言,UDDI 注册提供的信息包含以下 3 个方面的内容。

① 白页:UDDI 注册者提供的基本信息,包括联系地址、联系人和相关的联系标识符。

② 黄页:根据标准分类法进行的行业类别划分。

③ 绿页:服务发布者提供的公开大众的服务技术信息,是服务使用者所需要的全部内容。

4. BPEL4WS

BPEL4WS(Business Process Execution Language for Web Services,Web 服务的业务流程执行语言,也被称为 BPEL 或 BPELWS)是 2002 年 8 月由 Microsoft、IBM 和 BEA 公司联合发布的一种使用 Web 服务组合,用于为分布计算或网格计算环境开启共享任务的基于 XML 的语言,是 IBM 的 WSFL(支持面向图形的流程)和 Microsoft 的 XLANG(支持流程的结构化构造)的结合物。BPEL4WS 经过升级、发展,已于 2007 年 4 月被 OASIS 正式批准为 BPEL 的最新标准。

作为可执行流程的实现语言,BPEL4WS 的作用是将一组现有服务集成为一个新的 Web 服务。相对于对象组装技术,服务组装更为复杂。在企业内部,BPEL 用于标准化企业应用程序集成以及将此集成扩展到先前孤立的系统;在企业之间,BPEL 使与业务合作伙伴的集成变得更容易,更高效。一般而言,BPEL 提供的服务组装模型提供了以下特性。

① 灵活性。服务组装模型应该具有丰富的表现能力,能够描述复杂的交互场景,而且

能够快速地适应变化。

② 嵌套组装。一个业务流程可以表现为一个标准的Web服务，并被组装到其他流程或服务中，构成更粗粒度的服务，提高了服务的可伸缩性和重用性。

③ 关注点分离。BPEL只关注与服务组装的业务逻辑，而其他关注点，比如服务质量、事务处理等，可被作为附加扩展，由具体实现平台进行处理。

④ 会话状态和生命周期管理。与无状态的Web服务不同，一个业务流程通常具有明确的生命周期模型。BPEL提供了对长时间运行、有状态交互的支持。

⑤ 可恢复性。BPEL提供了内置的失败处理和补偿机制，对可预测的错误进行必要的处理。

常见的BPEL引擎主要有瑞士Berne大学的Bexee，EndPoint公司的ActiveBPEL以及Twister等。

5. Web服务开发环境

目前针对UDDI协议体系有许多实现的产品，其中提供了比较完整的Web服务开发环境和开发工具的产品主要有微软公司的Visual Studio .NET，IBM公司的WSDE(XML and Web Service Development Environment)&WSTK(Web Service Toolkit)和IONA公司的iPortal XMLBus等，相比较而言，Visual Studio .NET是在选择微软Windows系列的运行环境和开发语言时的一种不错的选择。而如果选择想跨平台的Java开发语言和基于Java EE的运行环境，则WSDE&WSTK、Netbeans、Eclipse甚至iPortal XMLBus等开发环境则更合适。

7.5 Web应用开发框架

如5.1.2节所述，框架是整个或部分系统的可重用设计，是一组可复用的设计组件，而且是经过良好测试的软件组件，易于扩展，规定了应用的架构，阐明了整体设计、协作组件之间的依赖关系、责任分配和控制流程，表现为一组抽象类以及实例之间协作的方法框架，可以被完全重用，也可以定制使用一部分。

将同类问题的解决途径进行抽象，把不同应用程序中的共性抽取出来实现为应用框架，即实现了某应用领域通用完备功能的底层服务。Web应用框架常常包括一些用于管理内容、访问控制系统和数据库接口、管理用户会话以及处理显示和风格的机制。使用Web应用框架的Web应用程序开发人员可以在一个通用功能已经实现的框架的基础上开发具体的应用程序，使开发者可以集中精力开发系统所需的商业逻辑，从而降低系统开发的难度，提高Web应用开发的效率，使系统的可扩展性、可维护性和灵活性得到提高。例如，在新闻系统中，采用了Struts、Sping和Hibernate三个开发框架，实现了一个典型的MVC架构。

7.5.1 Java EE开发框架

基于Java EE的Web应用开发的框架很多，常用的有Spring、Struts、Hibernate、WebWork等，本节简要介绍几种框架。

1. SSH(Spring、Struts 和 Hibernate)

SSH 是 Spring、Struts 和 Hibernate 的结合，是目前基于 Java EE 的企业级 Web 应用开源框架。

1) Spring

Spring 框架是为了解决企业应用开发的复杂性而创建的开源框架，主要优势之一就是其分层架构。分层架构允许在选择使用哪一个组件的同时，为 J2EE 应用程序开发提供集成的框架。Spring 框架易于使用并整合各类框架，能统一配置和部署，灵活并且可扩展，测试简单，开发成本低。

Spring 框架依赖注入机制，可以在运行期间为组件配置所需资源，而无须在编写组件代码时就加以指定，从而在相当程度上降低了组件之间的耦合性。客户代码仅仅面向接口编程，而无须知道实现类的具体名称。同时，开发人员可以很简单地通过修改配置文件来切换具体的底层实现类，自定义组件并不需要实现框架指定的接口，易于将组件从 Spring 中脱离，甚至不需要做任何修改，方便转移到其他框架中去，使得组件间的依赖关系减少，极大改善了代码的可重用性。

2) Struts

Struts 是 Apache 软件组织负责开发的一个基于 MVC 模式的企业级 Web 应用架构的开源框架，主要采用 Java Servlet、JSP 和 XML 等技术来实现。Struts 把 Servlet、JSP、自定义标签和信息资源(Message Resources)整合到一个统一的框架中，使利用者在开发时减少自己的编码量，节约开发时间。

Struts 中使用 PlugIn 实现 Web 应用设计架构中的接口层，ActionServlet、Action 和 ActionForm 等类实现抽象层，后台实现各种实现层的功能。Struts 框架中的模型、视图和控制器 3 部分是通过 struts-config. xml 配置文件将其联系在一起。

采用 Struts 框架可以加快开发速度，增强系统灵活性并降低系统的耦合性；使用标记能够把逻辑处理代码从页面中分离出来，简化页面开发；提高代码的重要率。

3) Hibernate

Hibernate 是开源对象关系映射框架，对 JDBC 进行了非常轻量级的对象封装，使 Java 程序员可以使用对象编程思维来操作数据库。Hibernate 查询语言(HQL)被设计成 SQL 的一个轻量级面向对象扩展，是对象和关系之间的桥梁。Hibernate 也支持用原始 SQL 或基于 Java 的标准和示例查询表达查询。Hibernate 使用 XML(. hbm. xml)文件把 Java 类映射到表，把 JavaBean 属性映射到数据库表。通过 JDBC 技术，支持所有的 SQL 数据库管理系统。

Hibernate 与 Java EE 应用程序服务器和 Web 容器都能很好地集成。

2. WebWork

WebWork 是由 OpenSymphony 组织开发的，致力于组件化和代码重用的拉出式 MVC 模式 J2EE Web 框架。现在 WebWork 已经被拆分成了 Xwork1 和 WebWork2 两个项目。Xwork 是一个标准的 Command 模式的实现，并且完全从 Web 层脱离出来，提供了前端拦截机(interceptor)、运行时表单属性验证、类型转换、强大的表达式语言(the Object Graph

Notation Language,OGNL)、IoC(Inversion of Control,控制反转)容器等核心功能。

WebWork2 建立在 Xwork 之上,处理 HTTP 响应和请求。WebWork2 使用 ServletDispatcher 将 HTTP 请求的变成 Action(业务层 Action 类)、session(会话) application(应用程序)范围的映射、request 请求参数映射。WebWork2 支持多视图表示,视图部分可以使用 JSP、Velocity、FreeMarker、JasperReports 和 XML 等。WebWork2.2 在 DWR 与 Dojo 这两个框架的基础之上,添加了对 AJAX 的支持。

3. Tapestry

Tapestry 框架是一个开源 Java Web 应用框架,是一个基于控件的框架。它位于 Java Servlet 容器和 Tapestry 应用程序之间,是 Servlet 的扩展,运行于如 Tomcat、JBoss、Websphere、WebLogic 等包含 Servlet 容器的应用服务器中。

Tapestry 框架提供了一个描述 Web 应用实现和由不同开发人员提供的元素之间交互的一致的方法。如导航条、嵌入的查询表格和登录按钮等元素,都以 Tapestry 组件的形式实现,单独测试,并在每一个页面上重用,保证了一致的外观和统一的交互行为。

4. JSF

JSF(Java Server Faces)是用于构建 Web 应用程序的标准 Java 框架,通过提供标准可扩展的用户界面组件、易配置的页面导航、方便的数据验证和转换、自动化 Bean 管理、事件处理、方便的错误处理以及内置对国际化的支持来加速 Web 应用开发。JSF 中,Model 是进行业务操作的部分,用来实现业务逻辑,一般使用 JavaBean 或 EJB 来建立复杂的企业应用;View 是由 JSF 标签构成的 JSP 页面组成,通过一个字符串松耦合到 Controller,通过发送事件来间接调用 Controller 的逻辑;Controller 主要包括 FacesServlet、配置文件和 action 处理器,其中 FacesServlet 负责从客户端接收请求,执行一组合理步骤(重建组件树,应用请求值,处理验证,更新模型值,调用应用,呈现响应)来准备和派发响应。

JSF 技术简化了 Web 应用用户界面的开发,各种技术水平的开发者都能够快速创建 Web 应用。通过在页面中装配一些可重复使用 UI 组件,并把这些组件与应用程序的数据源连接起来,也可把客户端产生的事件与服务端事件处理者连接起来。

JSF 框架包括以下两类主要组件:用于 UI 组件管理、UI 状态管理、事件处理、输入验证、页面导航、国际化以及可访问性的 Java API 和用于表示 JSP 页面中的 JSF 界面的一个 JSP 自定义标签库。

5. Turbine

Apache Turbine 是一个基于 Servlet 的开源 Web 应用框架,提供可定制 Web 应用的开发包,它使得开发人员可以快速、安全地构建基于 Java EE 的 Web 应用。其完整的体系结构包括:Turbine 核心框架,JDO 的 Turbine 实现 Torque,服务框架 Fulcrum,Maven 更新工具,基于 POM(项目对象模型)概念的 JCS(Java Caching System,Java 缓冲系统),Turbine 和 Avalon 的移植工具 Stratum 以及 Web 模板项目 Velocity。

Turbine 以 Servlet 作为基础框架,支持多种展示层技术。它类似于 Struts,但是它并没有与 JSP 耦合,它的特点是提供了大量可重用的离散的组件。Turbine 有三种使用方式:

①作为MVC架构的中心控制器；②作为其他应用使用的组件，Jetspeed门户框架就是基于Turbine而开发；③作为对象关系工具。

Turbine作为MVC架构的控制器时，由Turbine Servlet和Action Event Handles完成。Turbine Servlet直接和客户接口交互，接收用户请求并给用户返回响应，管理系统的业务流程、View和Model。Action表示对业务数据进行哪些处理，Action Handle针对不同的Action而做出处理。

Turbine可以和EJB集成，把EJB作为Web应用的业务处理层，在这种情况下，Turbine将成为EJB的前端。开发人员需要创建新的Turbine Service访问EJB系统。

6. Maverick框架

Maverick是一个轻量而完备的MVC Model 2框架。Maverick的Action充当Controller。Controller只接受一个ControllerContext参数。Request、Response、Servlet config和Servlet context等输入信息都包装在ControllerContext里面，而且Model也通过ControllerContext的model属性返回，整个编程结构清楚。但由于ControllerContext只有一个model属性可以传递数据，程序员必须把所有需要的数据都打包在一个对象里面设置到model属性里。

7. OPS

OPS(Orbeon Presentation Server)是一个开放源代码的基于Java EE平台且以XML为中心的框架。OPS是围绕XHTML、Xforms、XSLT、XML pipelines以及Web服务进行构建的。可以利用OPS来开发检索、处理和表达XML数据的应用程序。不像其他流行的Web框架，如Struts或WebWork(它们都是基于Java对象与JSP的)，OPS是基于XML文档和XML技术的。这种结构将为处理、表达和检索以XML为格式的信息提供一个更好的方案，并且几乎在实现表示层的时候不需要编写任何Java代码。

7.5.2 .NET框架

.NET框架是由微软开发的一个致力于敏捷软件开发、快速应用开发、平台无关性和网络透明化的软件开发平台，是一个可以构建、发布以及运行Web服务及其他应用程序的环境，它提供了托管执行环境、简化的开发和部署以及与各种编程语言的集成。

.NET框架的两个关键组件为CLR和.NET框架类库，其中，.NET框架类库包括ADO.NET、ASP.NET、Windows窗体和Windows Presentation Foundation (WPF)。

CLR运行代码并提供使开发过程更轻松的服务。CLR的概念类似于Java虚拟机(JVM)，其功能通过编译器和工具公开，开发者可以编写利用此托管执行环境的代码。也提供常用的自动内存管理(垃圾收集)、线程管理、反射、类型检查、异常管理等提高开发者开发效率的机制。使用C#、C++/CLI、VB.NET等基于CLR的语言编译器开发的代码称为托管代码。托管代码具有许多优点，例如，跨语言集成、跨语言异常处理、增强的安全性、版本控制和部署支持等。例如，用C#开发的类库可以被VB.NET调用，反之亦然。这种跨语言集成之所以成为可能，是因为基于CLR的语言编译器和工具使用由CLR定义的公共类型系统(Common Type System)，而且它们遵循CLR关于定义新类型以及创建、

使用、保持和绑定到类型的规则。另外，CLR也支持托管和非托管代码之间的互操作性，托管代码可以继续使用现有的COM组件和DLL等。CLR不仅可以在Windows中运行，还可由SQL Server或IIS承载，这也为开发数据库和网络应用程序提供了很大的便利。

.NET框架类库是一个与CLR紧密集成的可重用的面向对象类型集合，包含大量的常见的数据类型和操作，如下所列。

① 对字符、字符串、文本的操作。

② 对Windows操作系统中磁盘、目录、文件、注册表的操作。

③ 对UTF等国际化编码的支持。

④ 对数组、枚举、接口、泛型等语法的支持。

⑤ 对SQL Server、Oracle等主流的数据库的支持等。

框架类库可以充分地降低将开发人员编程的难度，让开发人员更为轻松地完成开发工作。另外，.NET框架不仅运行于个人计算机上，也运行在Windows Phone智能手机上。基于.NET框架开发的软件可以方便地迁移到Windows Phone智能手机上，这也降低了开发者的学习成本和工作量。

7.5.3 Web层开发框架

1. WebPage 3.0

WebPage 3.0是基于组件的、可视化的、轻量级的Web开发框架，基于标准技术，有极好的稳定性和可扩展性。WebPage 3.0基于MVC模式，重点关注View部分，达到可视化开发和最大限度的重用。主要使用Java、JSP、Servlet、HTML、JavaScript和XML等技术。WebPage 3.0开放组件设计接口，可以自由开发能在WebPage 3.0中使用的组件，而且开发组件非常简单。WebPage 3.0能大大提高Web层的开发速度。

2. Flex框架

随着Adobe Flex及其相关技术正成为RIA领域的主流技术，许多开源MVC框架也因此被开发出来，常用的有以下几种。

1）Cairngorm

Cairngorm是由Adobe公司推出的一个轻量级的Flex RIA程序开发框架，以提高程序的可扩展性、可维护性。早期的Cairngorm并不是一个完整的企业应用，它只是提供了一个开发骨架，Adobe称为体系。在提高程序可扩展性和可维护性的同时，早期Cairngorm也付出了相应的代价，集中表现为异常繁琐的文件书写，开发人员为了完成一个简单的功能往往需要修改许多文件。

为了解决上述问题，Cairngorm升级为Cairngorm 3.0。这一升级使得Cairngorm 3.0成为了一个真正的MVC框架，总体来讲，Cairngorm 3.0已与Java的Spring框架相类似。它的MVC分层结构有助于开发人员开发出具有复杂业务逻辑的可伸缩的RIA应用。在异步通信、事件驱动、无线程的Flex平台上，Cairngorm为构建商业应用程序提供了快速而可靠的方法。

2) PureMVC

PureMVC 是一个轻量级的使用 ActionScript 3 进行程序开发的框架，基于 MVC 模式构建而成。与 Cairngorm 不同的是，PureMVC 并不依赖于 Flash、Flex 或者 AIR 的特别类，因此特别适合作为 ActionScript 3 应用程序的架构。又因为 PureMVC 是用 ActionScript 实现的，而 ActionScript 与 JavaScript 均为基于 ECMAScript 标准的完全面向对象的语言，因此 PureMVC 较容易移植到其他开发平台上。

3) Model Glue:Flex

Model Glue:Flex 也是一个简单的 MVC Flex 框架。它的设计意图是成为一个比 Cairngorm 和 PureMVC 更轻量级的框架，同时它还拥有 ColdFusion 接口，因此它相比以上两种架构更为轻巧，更便于快速原型和简单的 Flex 应用开发。

4) Foundry

ServeBox Foundry 是为中大型规模的企业 RIA 系统开发而设计的 ActionScript 3 Java 框架，它包含了 Java 公用模块，便于将 Foundry Flex 插件与服务器端的进程紧密集成。同时它还可以解决 Flex 2 开发中一些常见且较为复杂的问题，例如 Model 与 View 同步、屏幕浏览、访问控制列表以及标签文字外部化等。

5) Guasax

Guasax 是一个易于使用的编程框架，它可以开发出条理清晰和可伸缩的 Flex 应用程序。Guasax 框架在运行时依照 MVC 模式来处理程序动作。Guasax 的一个独特之处是它用一个 XML 文件来配置业务逻辑中的动作，在某些方面类似于 Java Struts 框架。

6) ARP

ARP(Ariaware RIA Platform)最初是作为一个 ActionScript 框架而开发的，经过不断发展现在它已成为 Open Source Flash 项目群的一员，支持使用 ActionScript 2 和 ActionScript 3 开发 Flash 和 Flex 的 RIA 应用。

7) Flest

Flest 是构建企业级 RIA 的 ActionScript 3 Flex 应用程序框架。它运用了 Controller、Factory 和 Command 等设计模式，遵循高效、简单和实用的设计目标。作为一个轻量级且易于使用的工具集合，Flest 不但可以帮助搭建开发环境，还可以给开发人员最大的自由去实现自己的设计目标。

Flex 框架还有如 EasyMVC、Adobe FAST 和 Joeberkovitz 等。在众多的 Flex MVC 框架中，目前使用较多的是 Cairngorm 和 PureMVC。Cairngorm 以事件驱动为核心进行解耦，偏向于网络应用方面，侧重于处理客户端状态、业务逻辑及服务请求，且使用简单，易于学习；PureMVC 以消息发送(Notification)方式来解耦，是一个纯粹的 MVC 框架，但相对较复杂。

3. AJAX 框架

AJAX 作为重要的 Web 2.0 技术，被广泛采用。随之也出现了很多 AJAX 框架，如 Prototype、jQuery、Mootools、DOJO、Ext JS 和 AFLAX 等。

1) Prototype

Prototype 是一个由 Sam Stephenson 创建的 JavaScript 类库，它定义了 JavaScript 的面

向对象扩展、DOM操作API和事件等。

Prototype的特点是功能实用而且规模较小，非常适合在中小型的Web应用中使用。开发AJAX应用需要编写大量的客户端JavaScript脚本，而Prototype框架可以大大地简化JavaScript代码的编写工作。同时，Prototype具备兼容各个浏览器的特性，使用该框架可以无须考虑浏览器兼容性的问题。

2）jQuery

jQuery是由美国John Resig在2006年创建的一种轻量级的免费JavaScript框架，它为Web的JavaScript开发提供辅助功能。

jQuery语法设计简单，使用方便，用户可以更加方便地处理HTML、DOM、Events，实现动画效果，并且提供对AJAX的支持。jQuery还兼容各种浏览器，同时文档说明详细，还有许多成熟的插件可供选择，逐渐成为目前最受欢迎的AJAX开发框架之一。

3）Mootools

Mootools是一款结构紧密的模块化JavaScript框架，最大特点是视觉效果和变换特性。Mootools是一个简洁、模块化、面向对象的开源JavaScript框架，它能够帮助用户更快、更简单地编写可扩展和兼容性强的JavaScript代码。

Mootools跟Prototype类似，语法几乎完全一样，而且功能更强大，可以实现优雅且可定制的动画，同时具有完整的API文档以及丰富的范例。

4）DOJO

DOJO是由JavaScript编写的开源工具包，组件丰富，是一个强大的面向对象JavaScript框架，其目的是简化动态Web应用开发。开发人员使用其底层API和兼容层编写兼容的JavaScript和简化复杂脚本。DOJO的事件系统、I/O API和通用语言增强形成了强大的编程环境。

DOJO的包加载机制和模块化的库结构，能保持更好的扩展性，提高执行性能，减轻了用户开发的工作量，并保持一定的灵活性（用户可以自己编写扩展）。DOJO目前支持IE、Firefox、Safari、Opera、Chrome、iOS和Android，已经是众多开源框架的选择，包括WebWork、Tapestry、Eclipse ATF、MyFaces。

5）Ext JS

Ext JS是由JackSlocum开发的一种主要用于创建前端用户界面，与后台技术无关的前端开源AJAX框架，用于使用AJAX、DHTML和DOM脚本构建富交互的Web应用，可用于.NET、Java和PHP等各种开发语言开发的应用中。

Ext JS是一个重量级的框架，它包含大量的UI元素。Ext JS的UI组件模型和开发理念成型于Yahoo组件库YUI和Java平台上的Swing，为开发者屏蔽了大量跨浏览器方面的处理。相对来说，Ext JS开发者直接针对DOM、W3C对象模型，开发UI组件轻松。

6）AFLAX

AFLAX是基于AJAX的“派生/合成”式（derivative/composite）技术，是一种用于Macromedia公司FlashTM平台的JavaScript库，使开发人员能够将JavaScript和Flash进行结合，创建出AJAX类型应用。

7.5.4 Ruby 框架

Ruby 出现后，也出现了多种 Ruby 框架，包括 RoR、Camping、Merb、Nitro、Ramaze、Sinatra 和 Bowline GUI 框架。本节简要介绍 Web 应用开发中常用的框架之一 RoR。

RoR(Ruby on Rails)是一个用于编写 Web 应用的框架，基于编程语言 Ruby。RoR 给 Web 应用开发人员提供了强大的框架支持。Rails 具有很好的灵活性、稳定性和可靠性，成本更低。Ruby 和 Rails 为 RoR 框架的开发提供了良好的支持，也为 Web 应用开发提供了一个新的开发工具。

RoR 使 Web 应用的开发人员有了一种新的选择，给开发人员带来的感觉不仅仅是一个开发工具。从架构上，RoR 是严格遵守 MVC 模式的开发框架，对开发人员具有很直观的强制性，可以使开发人员将注意力集中到业务逻辑的设计与实现上。在开发效率上，RoR 的开发效率的优势是非常明显的，可以比 Java EE 写出更少的代码(这一点来自于 Ruby)，并且省去了很多 XML 配置文件的时间和步骤。在数据库的配置上，RoR 的 ORM 不需要很多的配置脚本，并且移植起来非常方便。

7.5.5 Python 框架

Python 框架包括 CherryPy、CubicWeb、Flask、Pylons 以及最新发布的 BlueBream1.0 等。本节简要介绍常用的 Zope 框架。

Zope 是一种让具备不同技能的开发人员一起构建 Web 应用的开源框架。Zope 涵盖了很多 Web 应用开发的底层细节，比如数据的持久性、完整性和访问控制等。Zope 可以充分利用所提供的服务来更快速地构建 Web 应用。Zope 可以使用 Python 语言来编写 Web 应用中的逻辑处理部分，当然也可以用 Perl。Zope 还提供两种方式，一种方式就像模板一样，来处理文本、XML 和 HTML 这样的数据；另一种是文本模板标记语言(DTML)和 Zope 页面模板(ZPT)。不同于基于文件的 Web 模板系统，比如 ASP 或 PHP，Zope 是高度面向对象的 Web 应用开发平台。

7.5.6 Web 服务开发框架

1. Axis

Apache Axis 是 Apache WebService 的一个开源子项目，是基于 XML 的 Web 服务架构，其核心是一个 SOAP 处理器，处理对象是 SOAP 消息，通过在处理器间传送一个 MessageContext 对象来完成对 Axis 消息的处理。MessageContext 含有处理请求消息或响应消息所需的所有数据，其中关键数据有一个请求消息、一个响应消息以及当前消息处理的所有元数据。每个处理器都可以访问 MessageContext 的所有数据，因此，每一处理器都可以使用修改请求消息及任何可能存在的响应消息。

Axis 用于开发包括客户端、服务器端和 SOAP Gatway 等各种应用。它包含了 Java 和 C++语言实现的 SOAP 服务器，以及各种公用服务及 API 以生成和部署 Web 服务应用。使用 Apache Axis 开发人员能够创建可互操作的、分布式的应用。

Apache Axis2 是 Apache Axis SOAP 项目的后继项目。此项目是 Web Service 核心引擎的重要改进，目标是成为 Web 服务和 SOA 的下一代平台。

2. .NET FrameWork

如 7.5.2 节所述，.NET FrameWork 是开发基于.NET 平台的 Web 服务框架。

3. XFire

XFire 是 codeHaus 组织提供的一种基于 Servlet 技术的开源 SOA 应用开发框架，构建了 POJO 和 SOA 之间的桥梁，主要特性是支持将 POJO 通过非常简单的方式发布成 Web 服务，不仅充分发挥了 POJO 的作用，简化了 Java 应用转化为 Web 服务的步骤和过程，也直接降低了 SOA 的实现难度，为企业转向 SOA 架构提供了一种简单可行的方式。

XFire 属于新一代的 Web 服务引擎，和其他 Web 服务引擎相比，XFire 的配置简单，可以非常容易地和 Spring 集成，它使得 Java 开发人员获得和.NET 开发人员一样的开发效率。

4. Apache CXF

Apache CXF(Celtix 和 XFire 的结合)是一个开源服务框架，继承了 ObjectWeb Celtix 和 Codehaus XFire 两个开源项目的精华，实现了 JCP 与 Web 服务中一些重要标准，简化了构造、集成 SOA 业务组件与技术的灵活应用。CXF 支持多种前端开发模型，通过编程 API (如 JAX-WS 和 JAX-RS）开发服务，这些服务支持多种协议，如 SOAP、XML/HTTP、RESTful HTTP 或 COBRA，可以在多种网络传输协议上运行，如 HTTP、JMS 或 JBI。

在 CXF 中，使用 WSDL 标准定义服务，并能够使用各种不同的消息格式和绑定方式。CXF 设计成可灵活部署到各种容器中，包括以 Spring 为基础的、JBI、SCA、Servlet 和 J2EE 容器。CXF 包含了大量的功能特性，其主要特性集中于如下几个方面。

1）支持 Web 服务标准。CXF 支持多种 Web 服务标准，包括 SOAP、Basic Profile、WS-Addressing、WS-Policy、WS-ReliableMessaging 和 WS-Security。

2）前端(Frontend)。CXF 支持多种前端编程模型，实现了 JAX-WS API（遵循 JAX-WS 2.0 TCK 版本)，也包含一个"simple frontend"允许客户端和终端(EndPoint)的创建，而不需要(Annotation 注解)。CXF 既支持 WSDL 优先开发，也支持从 Java 代码优先开发模式，这是 CXF 的一个巨大特点，使开发者可以根据实际项目的需要而采用代码优先或者 WSDL 优先实现 Web 服务的发布和使用。

3）容易使用。CXF 设计得更加直观与容易使用，有大量简单的 API 用于快速构建代码优先的服务，各种 Maven 的插件也使集成更加容易，支持 JAX-WS API，支持 Spring 2.0 以及更加简化的 XML 配置方式，等等。

4）支持二进制和遗留协议。CXF 是一种可插拨的架构，既支持 XML，也支持非 XML 的类型绑定，比如，JSON 和 CORBA。

5. ActionWebService

ActionWebService 是 Ruby on Rails 平台上的 Web 服务框架，支持 SOA、XML-RPC

以及动态生成 WSDL 协议的 API。客户端和服务器端使用相同的 API 定义，与其他基于 Action 的 Web 服务应用之间易于交互。

6. Python Web 服务框架

Suds、Zolera 和 Soaplib 是支持 SOAP、WSDL 协议的 Python 平台的 Web 服务框架。Suds 的消息模型是客户端，Soaplib 的消息模型是服务器，而 Zolera 的消息模型是客户端和服务器端两者。

7.5.7 Web 应用开发框架的选择

从前面几节中就可以看出，Web 应用开发框架种类繁多，开发人员需结合自身的实际项目需求与应用情况，选择合适自己的开发框架，这一点非常重要。只有针对自身的实际情况，找到适合项目当前的框架，才能取得事半功倍的效果。在选择开发框架的过程中需要考虑以下一些原则。

① 选择能够对开发过程提供更多、更好帮助的 Web 应用开发框架。
② Web 应用开发框架的学习要简单，上手要快。
③ 要有很好的技术和文档支持。
④ Web 应用开发框架结合其他技术的能力要强。
⑤ Web 应用开发框架的扩展能力要强。
⑥ Web 应用开发框架最好能提供可视化的开发和配置。
⑦ Web 应用开发框架的设计结构要合理。
⑧ Web 应用开发框架要运行稳定，运行效率高。
⑨ Web 应用开发框架要能很好地结合目前公司的积累。
⑩ 注意判断应用的场景和开发框架的适用性。

7.6 Web 应用构建工具

Web 应用开发工具，主要是针对 Web 应用的源代码的编写。在 Web 应用开发过程中，好的开发工具可以大大提高开发效率，节省开发人员的劳动量，从而大大提高 Web 应用的质量。“工欲善其事，必先利其器”，选择一个好的开发工具很有必要。如由于新闻系统属于 Java EE 应用，所以采用 Eclipse 作为开发工具。以下介绍几种常见的 Web 开发工具。

1. Visual Studio

Visual Studio 是微软公司推出的开发环境，是目前最流行的 Windows 平台应用程序开发环境。目前已经开发到 10.0 版本，也就是 Visual Studio 2010。Visual Studio 可以用来创建 Windows 平台下的 Windows 应用程序、Web 应用和 Web 服务，也可以用来创建智能设备应用程序和 Office 插件等。

2. Eclipse

Eclipse 是一个开放源码项目，是 Visual Age for Java 的替代品，其界面与先前的

Visual Age for Java类似，但由于其开放源码，任何人都可以免费得到，并可以在此基础上开发各自的插件，因此越来越受人们欢迎。

Eclipse是一个开源的、基于Java的可扩展开发平台，它本身是一个框架和一组服务的集合，众多插件的支持使得Eclipse拥有其他功能相对固定的IDE软件很难具有的灵活性。虽然大多数用户很乐于将Eclipse当作Java IDE来使用，但Eclipse的目标不仅限于此。Eclipse还包括插件开发环境(Plug-in Development Environment，PDE)，这个组件主要针对希望扩展Eclipse的软件开发人员，因为它允许他们构建与Eclipse环境无缝集成的工具。由于Eclipse中的每样东西都是插件，对于给Eclipse提供插件，以及给用户提供一致和统一的集成开发环境而言，所有工具开发人员都具有同等的发挥场所。Eclipse插件功能还支持Web服务开发，如WTP(Web Tools Platform)，使开发者便于创建、发布、发现和使用服务。

MyEclipse是针对Eclipse平台的一组支持Java EE开发的插件，MyEclipse企业级工作平台(MyEclipse Enterprise Workbench，简称MyEclipse)是对Eclipse IDE的扩展，在数据库和Java EE的开发、发布以及应用程序服务器的整合方面极大地提高工作效率。它是功能丰富的Java EE集成开发环境，包括了完备的编码、调试、测试和发布功能，完整支持HTML、Struts、JSF、CSS、JavaScript、SQL和Hibernate。

3. NetBeans

NetBeans是由原Sun公司在2000年建立的开放源代码的软件开发工具，是一个开放的框架和可扩展的开发平台。NetBeans是一个功能齐全的开放源码Java IDE，可以帮助开发人员编写、编译、调试和部署Java应用，并将版本控制和XML编辑融入其众多功能之中。NetBeans支持Java SE、Java EE以及Java ME应用的开发，以及用于Web应用的API及软件的核心组件的创建。所有这些都为Java开发人员创造了一个可扩展的开放源码、多平台的Java集成开发环境，以支持他们在各自所选择的环境中从事开发工作，如Solaris、Linux、Windows或Macintosh等。

NetBeans中还包括JavaScript编辑器、Ruby & Rails工具设计、jMaki、PHP和JavaScript调试器预览。根据NetBeans的不同配置，支持不同应用程序的开发。如在NetBeans Ruby配置中绑定了GlassFish v3 Ruby运行库，而在NetBeans的Web/JavaEE和Full配置中，绑定GlassFish v2和GlassFish v3两个中间件。

4. JBuilder

JBuilder是Borland公司推出的基于组件的、可升级的和可视化的Java开发工具，使用JBuilder可以快速有效地开发各类Java应用，它使用的JDK与Sun公司标准的JDK不同，它经过了较多的修改，以便开发人员能够像开发Delphi应用那样开发Java应用。

JBuilder的核心有一部分采用了VCL技术，使得程序的条理非常清晰，对于初学者而言，也能完整地看完整个代码。JBuilder的另一个特点是简化了团队合作，它采用的Internet工作室技术使不同地区，甚至不同国家的人联合开发一个项目成为了可能。

5. WebDB

WebDB是Oracle公司推出的一款基于数据库的Web应用开发工具，是一种管理数据库对象、开发Web组件以及建立、配置和管理Web应用的开发工具包，提供了将Oracle数据库引入到Internet的简便、快捷的解决方案。

通过该工具所提供的各种向导，辅以PL/SQL、HTML或JavaScript语言，用户不仅可以通过浏览器访问和操纵Oracle数据库，而且可以方便快捷地开发出美观且功能完善的Web页面，并最终建立内容丰富、管理简单的Web应用。

6. Zend PHP Studio

Zend Studio是开发人员在使用PHP整个开发周期中唯一的集成开发环境(IDE)，它包括了PHP所有必需的开发部件，具备功能强大的专业编辑工具和调试工具，支持PHP语法加亮显示，支持语法自动填充功能，支持书签功能，支持语法自动缩排和代码复制功能，内置一个强大的PHP代码调试工具，支持本地和远程两种调试模式，支持多种高级调试功能。通过一整套编辑、调试、分析、优化和数据库工具，Zend Studio加速开发周期，并简化复杂的应用方案。

7.7 Web应用部署

Web应用的部署是把已经构建完成的Web应用发布到服务器上，通过对Web应用的运行环境进行配置，使得用户可以通过Internet访问该Web应用。Web应用的部署一般包括打包、发布和评估。

1. 部署粒度

因为Web应用的开发是增量迭代进行的，所以部署也会在Web应用发布的过程中发生很多次。

Web应用发布以一种非常细粒度的方式完成，在测试完某个组件后，将新组件从预发布服务器再发布到运行服务器。例如，发布对内容或功能的更正、演化列表中的重要新条目，或发布对已经部署了的Web应用的用户“现在”就要求的新内容或功能。

在很多情况下，这种细粒度的发布方式可能并不合适。如果有很多不断发生的小变化，用户可能会困惑，而且集成错误的可能性和副作用也会增加。因此，除非一个变化更正一个破坏性的错误时进行这种细粒度的发布外，其他在更新或变化影响不大的情况下，最好把一组变化打包进行发布。

每一个打包发布周期都为最终用户提供了一个具有可用功能和特性的Web应用增量。每个评估周期都为Web应用团队提供重要指导，并为下一个增量做出内容、功能、特征和方法的修改。

Web应用的部署不是一气呵成的，也需要经历一个渐进的过程。Web应用不断升级的特点，使Web应用升级总是伴随着新旧版本兼容问题的风险、用户使用习惯突然改变而造成用户流失的风险、系统宕机的风险。为了避免这些风险，Web应用的部署可以采用灰度

发布的策略，其主要思想是把影响集中到一个点，然后再发散到一个面，出现意外情况后很容易就回退，然后把收集到的用户反馈作为 Web 应用修改的依据对 Web 应用需要修改的地方进行改进。

2. 部署原则

一个 Web 应用增量的部署代表 Web 工程项目的一个重要里程碑。一个 Web 应用的部署为最终用户提供了内容和功能以及立竿见影的收益，但是如果部署的计划不好，错误很多，而且执行效率不高，那么用户会很不满意。为了确保 Web 应用的顺利部署，Web 应用项目团队在准备交付一个增量时，应该遵循如下一些关键原则。

① 管理客户对 Web 应用增量的期望。管理客户期望的首要问题是与客户进行有效的沟通，客户通常并不希望看到团队承诺交付而又没有实现的内容，Web 应用项目团队不应该轻易对客户做出很可能完不成的承诺，敏捷方法将便于内容和功能的增量交付，为客户提供 Web 应用更早的反馈，因此支持对客户期望的管理。

② 安装与测试交付包。在部署前，需要在不同的硬件平台、不同的浏览器配置和网络带宽以及不同的安全性设置下进行完整的测试。

③ 交付前建立支持制度。用户往往期望在出现问题时得到及时的反应和精确的信息，所以应该计划支持，准备支持过程和响应，并且建立合适的记录保持基质，这样 Web 应用团队才能够对支持请求的种类进行分类评估。

④ 先改正有缺陷的 Web 应用，然后再交付。出于时间的考虑，一些 Web 应用开发人员会交付一些质量很低下的增量，这样是不妥的。Web 应用一方面为用户带来了巨大的便利性，提高了用户的工作效率；另一方面也为 Web 应用项目团队提供一些有用的反馈。当一个增量投入使用后，应该鼓励最终用户对 Web 应用的特征、功能等方面的内容进行客观的评价，Web 应用开发人员收集并记录这些信息，及时地调整自身的开发计划以及改正 Web 应用中存在的问题与隐患，进而更高效地开发出用户期望的产品。

3. 部署环境

Web 应用发布的软件系统主要包括操作系统软件和 Web 应用服务器软件。目前，几乎所有的操作系统都可以支持 Web 应用的发布，常用的用于 Web 应用发布的操作系统包括 Windows Server 系列、Linux 和各类 UNIX 等。Windows 家族产品中的其他操作系统如 Windows XP 也支持一定的 Web 应用发布能力，但一般只用作平时的开发测试及学习。可进行 Web 应用发布的服务器软件有 Apache、IIS、Websphere 及 WebLogic 等，其中，除 IIS 只能适用于 Windows 操作系统外，其他软件都可跨平台使用。

4. 版本控制和 CMS

版本控制和内容管理系统（CMS）在 Web 应用构建和部署活动中起着重要的作用。由于 Web 应用变更快且频繁，所以要使 Web 应用构建和部署成功，就必须在变更管理工具的辅助下管理好变更。版本控制和 CMS 的优点如下。

① 版本控制和内容管理系统都是变更管理工具，而且和其他内容一致。

② 确保所发布的内容正确，而且和其他内容一致。

③ 控制和跟踪内容的变更，包括对谁可以做出变更进行强制的实现机制。

④ 检验实现了一个功能的正确版本及其和相关功能的版本的正确对应。

⑤ 使Web工程团队在系统失败或崩溃时，能快速重建Web应用。

⑥ 允许团队在最新版本遇到严重的、未预见的错误时，回滚到前一个版本。

随着Web应用的规模和复杂性的增加，构建和部署的每一步的全面版本控制和健壮CMS的要求也在增加。

7.8 总结与展望

选择合适的技术是开发成功Web应用的关键因素之一，开发人员需要从各个方面了解各种技术的特性，进而更好地使用它们。开发一个完整的Web应用经常需要明白各种不同的技术如何相互影响。

本章主要分析了Web的开发与部署。Web应用开发涉及到Web应用通信协议、客户端开发技术、服务器端开发技术、开发框架、开发工具等内容，选择合适的开发技术和开发框架可以提高Web应用开发的效率和质量。而Web应用的发布与部署的粒度特性，使其需要遵循一些原则，了解部署环境，做好版本和内容的管理与控制。

很难说未来Web工程中会占据主导地位的技术是什么，不过Web技术一定会像更强大和更易于使用的方向发展。由于Web服务很大程度上依赖于XML标准，所以XML以及相关技术会在未来的Web工程中有很大的可能性被使用。而在客户端开发技术方面，随着微软称IE 9全面支持HTML5，未来几年HTML5以及CSS3将会迅速成为主流。HTML5将改变描述页面的方式，成为通往语义Web的重要阶梯。CSS3会让Web内容的展示变得更加容易。而到底采用什么技术，需要根据技术的特点和所要开发的Web应用的需求来选择确定。

第8章 Web应用测试

在市场需求的推动下，Web应用已从一般的Web应用发展成为大型电子商务、信息发布和提供各种服务的平台，规模不断扩大，复杂性日益增加，但激烈的商业竞争使得本来开发周期就很短的Web应用的开发周期更短，因此，如何保证Web应用的正确性和可靠性越来越成为人们关注的焦点。完善的Web应用测试是保障Web应用高可靠性和高可用性的必要手段，如何进行测试成为日益迫切的问题。与此同时，随着需求量与应用领域的不断扩大，对Web应用的正确性、有效性、可用性和对Web服务器等方面都提出了越来越高的要求。Web应用测试是最重要的质量保证措施之一，因此如何对Web应用进行有效地、系统地测试已经成为了Web应用开发的重要工作之一。

Web应用的测试、确认和验收是一项重要并且富有挑战性的工作，它对测试人员也提出了更高的要求。Web应用测试不但需要检查和验证Web应用是否按照设计的要求运行，而且还要测试Web应用在不同客户端的浏览器上运行是否正确，显示是否合适，Web应用是否符合用户的使用习惯，Web应用是否能够满足并发量的要求等内容。更重要的是，还要从最终用户的角度进行安全性和可用性等方面的测试。然而，Internet和Web媒体的不可预见性使得测试Web应用变得非常困难。因此，必须制作详细的测试计划，设计切合实际的Web应用测试规程，认真仔细地执行测试操作，才能最大限度地发挥Web应用测试的效能，尽可能地发现Web应用中存在的问题，最大限度地提高Web应用的质量。

根据测试内容的不同，Web应用测试主要包括功能测试、内容测试、性能测试、Web页面测试、客户端兼容性测试和安全性测试等内容。

8.1 Web应用测试特性

Web应用测试与传统软件测试一样，主要目的是发现错误和缺陷。Web应用具有多层体系结构，客户、数据通信、硬件以及服务器之间依赖关系非常复杂，每层内以及各层之间都有可能发生故障。如客户端浏览器的版本、型号的不同，所采用的页面渲染技术也不同，使得有些信息往往不能正常显示，从而产生兼容性问题和显示故障；服务器、数据库的负载能力有限，在用户访问达到高峰时，响应时间太长甚至不接受用户的访问；等等。要发现、分析进而排除这些故障，通常需要进行多方面的测试，以保证Web应用具有良好的功能、性能、兼容性和安全性。

从测试角度看，与传统的软件测试相比，Web 应用测试的特性和面临的挑战主要表现在以下一些方面。

(1) 用户数量巨大，并要求能提供对 Web 资源的跨平台全局访问，需要有处理并发事务的能力，因而需要进行多用户访问的性能测试。

(2) "内容"中的错误，通常只能靠人工完成；超文本结构中的页面之间链接关系要确保正确，面对很大挑战；展示层的"软"需求，如艺术性，难以测试；全球性使得 Web 应用可用性和多语言方面的测试难度更大。

(3) Web 应用一般采用多层架构，因此需要对测试结果进行综合分析，以确定系统功能或性能缺陷存在的具体位置。

(4) Web 应用中常常集成不同的软件，如 Web 服务器、数据库、中间件服务器等，以及第三方软件和框架。Web 应用的质量受所有这些第三方软件和组件及其交互的质量的影响，测试也必须包括对第三方软件和组件及其集成和配置的测试。

(5) Web 浏览器提供的导航，如回退(Back)、前进(Forward)和刷新(Refresh)等按钮，也会经常引发各种错误。

(6) 实现 Web 应用的不同种类技术的出现，使得以适当的粒度定义和验证 Web 测试模型的各个组件变得非常困难，充分建立组件间的关系也是一个大的难题。

(7) 从运行机制上看，Web 应用具有分布式、动态性、多平台、交互式和超文本等特点，运行环境异构、自治，要求针对其特性分别选择测试方法。为降低测试成本，提高测试效率，需要一定的辅助工具以提高测试执行的自动化水平。而 Web 应用测试方法和工具的不成熟性，使得测试面临错误、难以使用等问题。

Web 应用测试的诸多特性，使得不能仅用传统的方式来对 Web 应用进行测试。在 Web 应用整个生存周期的各个阶段，测试的侧重点有所不同。设计阶段测试的主要任务是：估算服务器端容量的规划是否合理，系统的安全设计是否合理，数据库设计是否合理，检查客户端设计的功能是否正确合理，检查系统的网络拓扑结构、容量设计是否合理。开发阶段测试的主要任务是：代码测试及组件测试，检查设计的代码能否满足规格需求。运行阶段测试的主要任务是：功能测试、性能测试、安全性测试、配置测试、兼容性测试及易用性测试。维护阶段测试的主要任务是根据维护的内容实施开发及运行阶段中的各个相关方面的测试。所有这些方面测试实施，对保证 Web 应用的质量及可靠性至关重要。

Web 应用的开发通常采用敏捷开发方式，可以不断调整以支持用户和客户多变的需求，并能够尽快投入市场。但这给测试带来了很大的压力：一旦 Web 应用发生了变化，必须进行回归测试，以便确定修改是否达到了预期的目的，检查修改是否损害了原有的正常功能，同时还需要补充新的测试用例来测试新的或被修改了的功能。

8.2 Web 应用测试过程

传统的测试方法将测试分为单元测试、集成测试、系统测试、接受测试和 Beta 测试，而过程包括计划、准备、执行和生成测试报告。传统测试方法定义工作结果，如质量计划、测试

测量、测试计划、测试用例、测试度量、测试环境和测试报告等，还定义测试角色，如测试经理、测试顾问、测试专家、工具专家等，还定义创建工作结果的详细步骤，如分析可用测试数据、准备或提供测试数据。

敏捷方法定义质量目标，然后根据团队自组织创建软件以满足这些质量目标。敏捷方法中开发团队协作和自治地寻找对问题的解决方案，测试不只是关于角色的，而是团队中紧密协作和可用能力的最佳使用。这说明，测试是一个集成开发活动，整个团队共同为质量负责，也因此共同进行测试。

对于Web应用测试而言，由于其开发周期短，虽然也需要完成这些不同层次的测试和执行测试过程，但并非在时间上顺序执行，而更倾向于尽早开始测试，迭代地测试小的增量，而且通常选择一些最重要的工作结果。

Web应用测试更多采用敏捷开发方法，进而很大程度上测试也采用敏捷方法。敏捷方法中单元测试用例由开发人员完成，其他测试工作由非开发人员完成。这就要求开发人员和测试人员之间进行有效地沟通和协调，相互理解，一旦有功能出现变化，这种变化可以立即被发现。另外，反馈、自动测试是短的开发周期和重构的前提。极限编程(XP)中结对编程、客户在场、持续集成和测试先行对Web应用测试和质量保证影响尤其大。特征驱动开发(FDD)并不使用结对编程，但是强调代码评审。但是这两种方法都确保在编码开始就采用静态质量保证技术。

一些研究者给出Web应用测试的一般过程：功能测试、内容测试和评审、Web页面测试、导航测试、接口测试、配置测试、安全测试、性能测试。测试主要包括以下几个步骤。

(1) 首先需要对被测试的Web应用进行需求分析，即对所做的测试做一个简要的介绍，包括描述测试的目标和范围，所测试的目标需要实现一个什么样的功能，总结基本文档、主要活动。

(2) 定义测试策略和方法，这里包括测试开始的条件、测试的类型、测试开始的标准以及所测试的功能、测试通过或失败的标准、结束测试的条件，测试过程中遇到什么样的情况终止和怎么处理，等等。

(3) 确定测试环境的要求(包括软件和硬件方面)，选择合适的测试用例、测试工具，决定执行测试的人员，以及确定测试要做到何种程度(测试的充分性)。

(4) 针对测试的行为，描述测试的细节，包括测试用例列表、进度表、错误等级分析、对测试计划的总结、测试过程会出现的风险分析等。

图8.1是Roger S. Pressman给出的Web应用测试任务。从图8.1中也可以看出，对一个Web应用进行测试，需要制定测试计划，以确保在给定的有限资源下顺利地完成所有的测试内容。

Web应用的测试，首先需要测试最终用户能够看到的内容和界面功能，随着测试的深入，再测试信息和功能架构以及导航方面的内容，最后测试的内容转到用户并不总是可见的测试技术能力，即Web应用基础设施及其安装与实现的问题。

Web应用测试

1. 评审利益相关者的需求。
    - 标识关键用户目标。
    - 对每类用户的用例进行评审。
2. 建立优先级，以确保每一个用户目标都将被适当地测试。
3. 根据要实施的测试类型的描述定义 Web 应用测试策略。
4. 制定测试计划。
    - 规定测试进度，并对每个测试分配职责。
    - 指定自动化测试工具。
    - 规定每一类测试的验收标准。
    - 详细说明缺陷跟踪机制。
    - 定义问题报告机制。
5. 进行"单元"测试。
    - 评审内容的语法和语义错误。
    - 评审内容的许可性。
    - 测试接口机制的正确操作。
    - 测试每一个组件（如脚本），确保正确的功能。
6. 进行"集成"测试。
    - 对照用例来测试界面的语义。
    - 实施导航测试。
7. 进行配置测试。
    - 评估客户端的配置兼容性。
    - 评估服务器端的配置。
8. 进行性能测试。
9. 进行安全性测试。

图 8.1　Web 应用测试任务

Web 页面测试验证用户界面的交互机制以及美学方面的内容，目的是发现由于实现了糟糕的交互机制而导致的错误，或者由于不小心而导致的遗漏、不一致或歧义性。

性能测试包括一系列测试，设计这些测试用来评估 Web 应用的响应时间及可靠性如何受增长的用户通信量和功能复杂性的影响，用来查找哪些 Web 应用组件与性能下降有关等内容。

安全性测试是将一系列测试合并起来，攻击 Web 应用及其环境中的弱点，目的是发现可能的安全漏洞。

8.3　功能测试

功能测试在整个测试过程中起着至关重要的作用，它结合 Web 应用规格说明的要求，保证 Web 应用在功能上能够达到预期的目标。只有满足功能需求的 Web 应用才可能是客户需要的，一个不满足功能性要求的 Web 应用可以称得上是无用的 Web 应用。根据测试内容的不同，功能性测试主要可以分为链接测试、表单测试、数据校验、Cookie 测试、数据库测试、应用程序特定的功能需求以及设计语言测试。

1. 链接测试

链接测试是Web应用所特有的测试，可分为三个方面的内容：首先，所有链接是否按指示的那样确实链接到了该链接所指向的资源；其次，所链接的资源是否存在；最后，是否所有页面都能够被链接到，不存在孤立页面（所谓孤立页面是指没有链接指向该页面，只有知道正确的URL地址才能访问）。

链接测试需要对整个Web应用的所有链接进行，而一般的Web应用内的链接错乱复杂，犹如一张大蜘蛛网，稍有疏忽便有测试不完全的地方，因此引入链接自动化测试能够大幅提高链接测试的效率。现在已经有许多工具可以帮助测试人员进行链接测试，如Xenu Link Sleuth和HTML Link Validator。链接测试必须在集成测试阶段完成，也就是说，在整个Web应用（或增量）的所有页面开发完成之后进行链接测试。

2. 交互测试

当包含表单元素的页面呈现给用户时，用户填写相关内容并提交到服务器，然后再由服务器端程序动态生成页面返回给用户，从而实现一次完整的交互过程。当用户使用表单元素进行各种操作时，必须校验用户提交给服务器的信息的正确性和有效性。例如用户填写的出生日期与职业是否恰当，填写的所属省份与所在城市是否匹配，等等。如果使用了默认值，还要检验默认值的正确性。如果表单元素只能接受指定的某些值，则也要进行测试。例如只能接受某些字符，测试时可以跳过这些字符，看是否会出现错误。也就是说，要测试这些程序，需要有数据正确性验证，对异常处理的验证，还需要验证服务器能正确解析和使用这些信息。

交互测试一般要确保以下几个方面的内容。

① 对表单元素中的标识域给出正确标记，并且为用户显式地标识出强制域。

② 对每项用户输入进行正确性和合法性检查。

③ 对用户输入错误时的异常处理机制进行检查。

④ 当用户没有从下拉菜单或按钮组中进行选择时，使用合适的默认项。

⑤ 浏览器“后退”等功能没有破坏输入到表单中的数据。

在更具体的层次上，测试应该确保：①表单元素有合适的宽度和数据类型；②表单元素建立了合适的安全措施，防止用户输入的文本字符串长度大于某个预先定义的最大值；③对下拉菜单中的所有合适的选项进行详细说明，并按照对最终用户有意义的方式排序；④浏览器“自动填充”特性不会导致数据输入错误；⑤Tab键或一些其他键能够使输入焦点在表单元素之间正确移动。

3. 数据校验

如果根据业务规则需要对用户输入进行一些正确性和合法性校验，需要保证这些校验功能正常工作，例如，省份的字段可以用一个有效列表进行校验。在这种情况下，需要验证列表完整而且程序正确调用了该列表，例如在列表中添加一个测试值，确定系统能够接受这个测试值。

在测试用户交互时，该项测试和交互测试可能会有一些重复。

4. Cookies 测试

Cookies通常用来存储用户信息和用户在Web应用中的操作。当一个用户使用

Cookies访问了某一个Web应用时，Web服务器将发送关于这一用户的信息，并把该信息以Cookies的形式存储在客户端计算机上，这可用来创建动态和自定义Web页面或者存储登录内容等信息。

如果Web应用使用了Cookies，就必须检查Cookies是否能正常工作。测试的内容可包括Cookies是否起作用，是否按预定的时间进行保存，刷新Web页面对Cookies是否有影响、有什么影响，等等。如果在Cookie中保存了注册信息，请确认该Cookie能够正常工作而且已对这些信息加密。如果使用Cookies来统计次数，需要验证次数累计正确。关于Cookies的使用可以参考浏览器的帮助信息。

可以通过采用IECookiesView等测试工具进行该项测试。IECookiesView是一个搜寻并显示出计算机中所有的Cookies档案的工具，它记录了是哪一个Web应用写入Cookies的，内容有什么，写入的时间日期及此Cookies的有效期限等信息，此软件只对IE浏览器的Cookies有效。

5. 数据库测试

数据库测试包括测试实际内容及其完整性，以确保数据没有损坏且模式正确。从功能角度而言，在使用了数据库的Web应用中，一般可能发生两种错误：数据完整性错误和输出错误。数据完整性错误主要是由于用户提交的表单信息不正确而造成的，数据完整性错误是使错误的结果被保存下来或者使字段、记录、表和数据库中数据失效的任何错误，例如可能漏掉了表中的记录，或者没有正确更新而导致数据过期，等等。而输出错误主要是由在数据提取和操作数据指令过程中发生的错误引起的。针对这两种情况，可分别进行测试。

对数据库功能的测试可以依赖于工具进行，常用的数据库测试工具有DBunit、QTP、DataFactory、Crash-me(MySQL自带的测试数据库的性能的工具)等。

6. 特定功能需求测试

功能测试中，有一点需要特别注意的是Web应用的特定的功能需求，测试人员需要对这些特定的功能需求进行测试。尝试用户可能进行的所有操作，例如在线购物中，涉及到的操作有：下订单，更改订单，取消订单，核对订单状态，在货物发送之前更改送货信息，在线支付，等等。这是用户之所以访问Web应用的原因，一定要确认Web应用能像广告宣传的那样神奇，达到其所宣扬的所有特定功能，否则，如果言过其实，用户会有受到欺骗的感觉，进而影响用户对Web应用的满意度。

进行特定功能需求测试的前提是测试人员必须深刻理解需求说明文档，否则，测试人员很难对这一测试有准确的把握。

8.4　内容测试

内容测试(和评审)试图发现内容方面的错误。Web应用内容中的错误可以小到简单的印刷错误，大到不正确的信息、不合适的组织，或者违背知识产权法。例如，在线购物网站上的商品价格列表中，错误的价格可能会引起财政问题甚至法律纠纷，新闻系统中虚假的新闻报道可能会引起不良的社会影响。信息的准确性是指页面文字表述是否符合语法逻辑或

者是否有拼写错误。内容测试试图在用户碰到这些问题及很多其他问题之前就发现它们。

这项测试活动在很多方面类似于对已写文档的审稿。大型 Web 应用会有专业审稿人员来发现排字错误、语法错误、内容一致性错误、图形展示错误、交叉引用错误。由于数据库系统已经在 Web 应用中广泛采用，因此，除了检查静态内容方面的错误，内容测试还要考虑从数据库系统所维护的数据中导出的动态内容。在很多情况下，这种形式的测试还需要在 Web 应用的生命周期中继续不断地增加新的内容。例如，对新闻系统而言，会有新内容不断地加入，这些内容需要被检查。

8.4.1 内容测试的目标

内容测试有三个重要的目标：①找出基于文本的文档、图形展示和其他媒体中的语法错误，例如，打字错误、语法错误；②找出当导航发生时所展现的任何内容对象中的语义错误，即信息的精确性和完备性方面的错误；③找出展示给最终用户的内容的组织或结构方面的错误。

为了达到第一个目标，可以借助自动拼写和语法检查工具，但不能完全依赖这些工具来完成任务，必须由内容测试人员人工发现。

语义测试关注于每一个内容对象所显示的信息方面。测试人员必须检查如下内容。

① 信息确实是最新而且准确的。

② 信息简洁扼要。

③ 内容对象的布局对于用户来说容易理解。

④ 嵌入在内容对象中的信息易于被发现。

⑤ 对于所有从其他地方导出的信息，提供了合适的引用。

⑥ 显示的信息内部一致，与其他内容对象中所显示的信息一致。

⑦ 内容不具有攻击性，不容易被误解，不会引起法律诉讼。

⑧ 内容不侵犯版权或商标。

⑨ 内容包括补充现有内容的内部链接，并检查正确性。

⑩ 内容的美学风格与界面的美学风格不相矛盾。

对于大型的 Web 应用来说，要检查所有这些方面是一项令人畏惧的任务。然而，如果有语义错误，则会动摇用户对 Web 应用的信任，并且会导致 Web 应用的失败。

在内容测试期间，要对内容架构进行测试，以确保将所需要的内容以合适的顺序和关系展现给最终用户。例如，幸福密码网显示了关于心理测试和干预等多种信息，干预和心理测试的类型之间有密切联系。内容架构的测试试图发现这种信息及其关联关系表示方面的错误。

8.4.2 验证动态内容

目前，绝大多数 Web 应用要与复杂的数据库管理系统连接，使用从数据库中获取的数据实时构建动态的内容对象。例如，有关证券交易的 Web 应用能够产生股票、基金等文本信息、表格信息和图形信息，当用户请求了某种股票信息后，就会自动创建表示这种信息的复合内容对象。完成此任务需要查询大型股票数据库，从数据库中抽取相关的数据，抽取的

数据必须被组织为一个内容对象，将这个内容对象(代表由某个最终用户请求的定制信息)传送到客户显示。这一系列步骤中，每个环节的结果都可能发生错误，并且一定会发生。数据库测试的目标是发现这些错误。然而，Web 应用的数据库测试会由于以下多种原因而变得复杂。

① 客户端请求的原始信息难以表示成能够输入到 DBMS 中的形式，如结构化查询语言 SQL，因此，需要测试将请求翻译成能够被这些 DBMS 处理的格式的过程，找出其中所产生的错误。

② 数据库可能离 Web 应用的服务器很远，因此需要测试 Web 应用和远程数据库之间的通信，找出所存在的错误。

③ 从数据库中获取的原始数据要传递给 Web 应用服务器，进行格式转换，以便随后传递给客户端，因此，应该测试 Web 应用服务器接收到的原始数据的有效性，测试转换的有效性，将这种转换应用于原始数据，能够生成有效的内容对象。

④ 动态内容对象要以能够显示给最终用户的形式传递给客户端，因此，应该测试内容对象格式，找出格式方面的错误、发现与不同的客户环境配置的兼容性问题。

考虑这 4 种因素，应用测试用例的设计方法，保证有效信息通过界面层在客户与服务器之间传递；保证 Web 应用正确地处理脚本，并且正确地抽取或格式化用户数据；保证用户数据被正确地传递给服务器端的数据转换功能，此功能将合适的查询格式化；保证查询被传递到数据管理层，本层与数据库访问程序通信。

通常，数据转换层、数据管理层和数据库访问层是使用可重用的组件来构造。这些可重用组件都分别进行了验证，因此，Web 应用的测试集中在客户层与 Web 应用和数据转换两个服务器层之间交互测试。

应该对用户界面层进行测试，确保对每一个用户查询都正确地构造了 Web 页面脚本，并且正确地传输给了服务器端。还应该对服务器端的 Web 应用层进行测试，确保能够从 Web 页面脚本中正确地抽取出用户数据，并且正确地传输给服务器端的数据转换层。应该对数据转换功能进行测试，确保创建了正确的 SQL，并且传给合适的数据管理组件。

8.5 Web 页面测试

Web 页面设计测试用于对 Web 应用设计的种种情况进行评价，包括设计中如何布局、如何搭配来体现用户至上的思想，指明操作的明确方向，提供反馈，维护语言和方法的一致性，等等，页面测试需要认真考虑包括易用性和外观这样的主观印象，也需要考虑像导航、自然流、可用性、命令和可访问性等内容。

1. Web 页面测试内容

Web 页面测试涉及的内容广泛，主要包括以下几个方面的内容。

(1) Web 应用地图和导航条位置是否合理，是否可以导航，等等；内容布局是否合理，滚动条等简介说明文字是否合理，位置是否正确。

(2) 背景/色调是否正确、美观，是否符合用户需求。

(3) Web 页面在窗口中的显示是否正确、美观(在调整浏览器窗口大小时，屏幕刷新是

否正确)，表单样式大小、格式，是否对提交数据进行验证(如果在页面部分进行验证的话)，等等。

(4) 链接的形式、位置是否易于理解等。

(5) Web页面元素清单，为实现功能，是否将所需要的元素全部都列出，如按钮、单选框、复选框、列表框、超链接和输入框等。

(6) Web页面元素的容错性(如输入框、时间列表或日历)是否存在，是否正确。

(7) Web页面元素基本功能是否实现，如文字特效、动画特效、按钮和超链接等。

(8) Web页面元素的外形、摆放位置是否合适，如按钮、列表框、复选框、输入框和超链接等。

(9) Web页面元素是否显示正确，主要针对文字、图形和签章等。

(10) 元素是否显示，是否存在。

Web页面包含元素众多，按照测试的侧重点不同，Web页面测试一般包括导航测试、图形测试、内容测试、表格测试、整体界面测试等。

2. 导航测试

导航描述了用户在一个Web页面内和在不同的用户接口控件之间操作的方式，例如按钮、对话框、列表和窗口等；或在不同的链接页面之间。导航测试要考虑下列问题，导航是否直观和易用；Web应用的主要部分是否可通过主页访问；Web应用是否需要站点地图，搜索引擎或其他的导航帮助等。

在一个Web页面上放太多的信息往往起到与预期相反的效果。Web应用的用户趋向于目的驱动，很快地扫描一个Web应用，看是否有满足自己需要的信息，如果没有，就会很快地离开。很少有用户愿意花时间去熟悉Web应用的结构，因此，Web应用导航帮助要尽可能地准确。导航的另一个重要方面是Web应用的页面结构、导航、菜单和链接的风格是否一致。确保用户凭直觉就知道Web应用里面是否还有内容，内容在什么地方。可以考虑让最终用户参与这种测试，效果将更加明显。

3. 图形测试

图形是Web应用页面中不可或缺的元素之一，是增强Web页面视觉吸引力的重要表现形式。适当的图形既能起到广告宣传的作用，又能起到美化Web页面的作用。一个Web应用的图形可以包括图片、动画、边框、颜色、字体、背景和按钮等。图形测试主要包括以下几个方面的内容。

(1) 确保Web应用中的图形都有明确的用途，图片或动画不要胡乱地堆在一起。Web应用的图片尺寸要尽量地小，并且要能清楚地说明某件事情，一般都链接到某个具体的页面。

(2) 验证所有Web页面的字体风格是否一致。

(3) 背景颜色应该与字体颜色和前景颜色相搭配。

(4) 验证图片的大小和质量是否合适。

(5) 需要验证文字环绕是否正确。不要因为使用图片而使窗口和段落排列古怪或者出现孤行。

(6) 验证图片是否能正常加载。

通常来说，使用少许或尽量不使用背景。如果需要使用背景，那么最好使用单色的，和导航条一起放在Web页面的左边。但是需要注意图案和图片可能会转移用户的注意力，所以不能过多的放置图片或图案。

4. 内容测试

如8.4节内容测试所述，内容测试是为发现内容方面的错误。在Web应用开发过程中，开发人员可能不是特别注重文字表述，有时文字的改动只是为了页面布局的美观，因此测试人员需要在Web页面测试时检查页面内容的文字表达是否恰当。

5. 表格测试

Web页面中经常会使用到表格，所以Web应用测试中需要验证表格的展示是否正确以及方式是否符合用户习惯等。比如测试时需考察用户是否需要向右滚动页面才能看见所关注的信息；每一栏的宽度是否足够宽；表格里的文字是否都有折行；是否有因为某一格的内容太多，而将整行的内容拉长；等等。

6. 整体页面测试

整体页面是指整个Web应用的页面结构设计，是给用户的一个整体感觉、第一印象，如果页面搭配不合理，整个页面给用户一个杂乱无章的感觉，再好的内容，也难以吸引用户。例如，当用户浏览Web应用时是否感到舒适，是否凭直觉就知道要找的信息在什么地方，整个Web应用的设计风格是否一致，等等。

对整体页面的测试过程，其实是一个对最终用户进行调查的过程。一般Web应用在主页上做一个调查问卷的形式，来得到最终用户的反馈，或者也可以通过发放问卷调查表的方式来获取用户对整体界面的印象。

对所有的用户页面测试来说，都需要有外部人员（与Web应用开发没有联系或联系很少的人员）的参与，最好是最终用户的参与，这样才能达到最好的测试效果。

7. 页面测试要素

表8.1列出了具体的需要测试的页面要素。

表8.1 Web页面测试的要素

要素类型	面对的问题
说明和技术信息	信息和指令的准确性
字体	样式一致性 字体清晰度 识别斜体和截线字体的困难度 一篇文档中多种字体造成的视觉混淆以及目标平台上字体的可用性
颜色	背景颜色的适宜性 前景颜色的适宜性 字体颜色的适宜性 精细、互补颜色的选择通常比饱和、反差色更悦目

续表

要素类型	面对的问题
边缘	命令按钮的三维效果对用户是有效的可视化提示 对非交互元素三维效果的使用会被混淆
图像	大图像可能增加装载时间 可视化提示和设计细节同背景是否能区分 背景的适宜性 标签清晰度 按钮清晰度 图片尺寸的适宜性
框架	浏览器能否正确显示 显示设置和浏览器类型影响框架的显示情况 回退按钮经常有意想不到的结果
表格	网格(表格中的表格)减慢 HTML 的装载速度 外观是否由于显示设置和浏览器类型而导致不正确的范围和重叠 测试应该包括所有浏览器、显示设置和浏览器窗口大小

8.6 兼容性测试

Web应用的运行环境的可变性和不稳定性是 Web 工程面临挑战的重要因素。由于用户群的不同，所以客户端的硬件设备、网络连接、操作系统、服务端支持、浏览器等各种因素都可能有所不同，这就导致了一个异构、自治的工作环境。不同的计算机、显示设备、操作系统、浏览器和网络连接速度等都会对 Web 应用产生巨大的影响。在某些情况下，小的兼容性问题显得并不是很重要，但是在大多数情况下，兼容性是一个非常严重的问题，忽略 Web 应用的兼容性，很可能导致用户无法使用该 Web 应用，从而直接降低 Web 应用的可用性。例如，缺少所需要的插件可能使得内容难以获取，浏览器的不同可能会较大地改变页面的布局，等等。

Web 应用兼容测试是测试 Web 应用在各种硬件、软件、操作系统、网络等不同的环境下的性能，Web 应用兼容性测试的目的是发现在主要的实际用户环境下(特定的客户或服务器环境中)运行程序时出现的错误。常见的 Web 应用兼容性测试有平台测试、浏览器测试、分辨率测试、连接速度测试、打印机测试、数据库兼容性测试和应用软件之间兼容性测试。一般通过将以上几种测试的某几种组合在一起对 Web 应用进行测试。根据实际情况，采取等价划分的方法，列出兼容性矩阵，然后判断待测试的内容是否能够正确或者符合习惯。

1. 平台测试

市场上有很多不同的操作系统平台，常见的有 Windows、Linux、Mac 等。而 Windows 操作系统又包括 Windows 7、Windows XP、Windows Vista、Windows 2000/NT、Windows 9x 等。Web 应用的最终用户究竟使用哪一种操作系统，取决于用户系统的配置。同一个 Web 应用可能在某些操作系统下能正常运行，但在另外的操作系统下可能会运行失败。

对于一些特殊项目(如定制项目),可以指定某一类型的操作系统版本,这些都应该在需求规格说明书中指明,针对这些指明的操作系统版本必须进行兼容性测试。而绝大多数的Web应用,是不指定操作系统版本的,针对这样的项目,应当针对当前的主流操作系统版本进行兼容性测试,在确保主流操作系统版本兼容性测试的前提下,再对非主流操作系统版本进行测试,尽量保证项目的操作系统版本的兼容性测试的完整性。

因此,在Web应用部署之前,需要在各种操作系统上对Web应用进行兼容性测试,例如Mac机就不兼容ActiveX。

2. 浏览器测试

浏览器是Web客户端最核心最直接的访问Web应用的工具。而由于浏览器生产商(Microsoft、Mozilla、Google、Opera 等)、版本(Internet Explorer 7、8 等)、操作系统(Windows Linux 等)和硬件设备(分辨率和颜色深度)等客户使用的浏览器环境一般都不同。由于缺乏统一的标准,不同环境下浏览器对同一个页面的展现或者行为可能也不同。

具体来说,来自不同厂商的浏览器对JavaScript、ActiveX、plug in或不同的HTML规格有不同的支持,即使是同一厂家的浏览器,也存在不同版本之间支持程度不同的问题。最典型的就是ActiveX,它是专为Internet Explorer设计的,不能在其他浏览器中使用。另外,框架和层次结构风格在不同的浏览器中显示也有不同,甚至根本不显示。因而要结合不同浏览器、操作系统和硬件设置进行Web应用测试,以观察在不同的客户硬件配置、客户操作系统、浏览器类型和版本以及浏览器插件的组合使用情况下,浏览器是否能正确显示所有元素,同时需要检查浏览器命令、内容设置、浏览器安全设置等是否满足要求。

3. 分辨率测试

分辨率测试是为了确保页面版式在不同的分辨率模式下能正常显示而进行的测试。

用户使用什么模式的分辨率,对于Web开发人员来说是未知的。通常情况下,在需求规格说明书中会建议用户使用某些分辨率。对于测试来讲,必须针对需求规格说明书中建议的分辨率进行专门的测试。现在常见的分辨率是1440×900、1280×800、1024×768等。对于需求规格说明书中规定的分辨率,测试必须保证测试通过,但对于其他分辨率,原则上也应该尽量保证不影响查看。

4. 连接速率测试

用户的网络环境各异,网速也相差很大,Web应用测试人员需尽可能考虑到各种网速情况下的页面加载情况,尤其是需要测试在网速较慢的情况下页面的加载速度。

5. 打印机测试

用户在使用Web应用的过程中,可能需要将Web页面上的重要信息打印下来,因此在兼容性测试的内容中需要包括打印机测试,来验证Web页面打印是否正常。

6. 数据库兼容性测试

现在Web应用大都是基于数据库系统的,对此类应用应测试对不同数据库平台的支持

能力，例如从 DB2 平台迁移到 SQL Server 平台时，Web 应用是否可直接链接，或者提供相关的转换工具。还需要测试新旧数据转换是否存在问题，Web 应用是否提供新旧数据转换的功能。例如，当 Web 应用升级后可能会定义新的数据格式或文件格式，这就涉及对原有格式的支持及更新，原有用户记录在新格式下是否依然可用，等等。另外，还需要测试转换过程中数据的完整性与正确性。

7. 应用软件间兼容性测试

应用软件间兼容性测试主要考察以下两项内容：①Web 应用运行需要哪些应用软件支持；②判断 Web 应用与其他常用软件一起使用，是否会造成其他软件运行错误或本身不能正确实现其功能。



8.7 性能测试

由于 Web 应用无法确定客户端用户对 Web 应用的访问情况，用户数目会处于不断的变动之中，而 Web 服务器的承载能力有限，并发数据量过大甚至超过服务器的承载能力，会导致 Web 应用阻塞甚至崩溃。

8.7.1 性能测试目标

Web 应用性能测试用于评估 Web 应用的响应时间及可靠性如何受增长的用户数量和通信量以及功能复杂性的影响，找出与性能有关的 Web 应用组件和特征，并确定性能下降如何影响 Web 应用的整体需求和目标。Web 应用性能测试是指在正常和疲劳使用被测 Web 应用的情况下，观察该 Web 应用的性能是否满足性能要求，确定 Web 应用的各项指标和参数，主要确定在用户可接受的响应时间内，能够承担的并发用户的数量、能够同时处理的业务的数目，以及不同负载情况下 Web 页面的下载时间和检查瓶颈可能发生的位置。Web 应用性能测试是确保 Web 服务器能够在规定的参数范围内响应浏览器的请求，是一种信息的收集和分析过程，过程中收集的数据用来预测怎样的负载水平将耗尽系统资源。

Web 应用通常采用多层架构，因此其性能受到以下几个因素的影响。

① 客户端配置(如浏览器缓存大小配置等)。

② 服务器端配置(如 RAM、CPU 个数、磁盘读写速度等)。

③ 网络通信容量。

④ 工作负载的类型及数量(如读取 Web 服务器上的不同类型的文件或实现服务器上数据库中的数据读写操作)。

在进行性能测试时，以下几个方面会影响性能测试的效果。

① 真实环境与测试环境差异较大。Web 应用的性能与运行环境有关，不同的环境配置可能导致不同的测试结果，而测试往往在一个局域网内或在企业 Intranet 内进行，这与真实的环境差异较大。

② 负载的不确定性。由于网络上并发的用户访问数随时发生着变化，同时每一个用户的行为又各不相同，例如表单提交行为、页面切换行为等的不同，直接导致负载种类及数量

的不同。

③ 模拟真实环境困难。模拟真实环境包括设备、配置、分布等模拟，实现起来难度很大。

④ 模拟真实用户行为困难。模拟用户行为主要体现在模拟用户的数量及行为，而这两方面均有主观性、随机性及不充分性。

性能测试的目的主要是为维护系统的性能，并找到有效的改善策略。然而由于Web应用综合了大量的技术，诸如HTML、CSS、JavaScript、Java、C#、PHP等，同时它还依赖很多其他的因素，比如连接、数据库和网络等，使得性能测试变得非常复杂。总的来说，Web应用性能测试的目的主要有如下几个。

(1) 评估系统性能。测试中得到的负荷和响应时间数据可以被用于验证Web应用的服务能力，并帮助做出决策。

(2) 确定系统瓶颈。受控的负荷可以被增加到一个极端的水平，并突破它，从而发现和修复系统的瓶颈或薄弱的地方。

(3) 系统调优。重复运行测试，验证调整系统的活动得到了预期的结果，从而改进性能。

(4) 检测Web应用中潜在的问题。长时间的测试执行可导致程序发生由于内存泄露引起的失败，揭示程序中的隐含的问题或冲突。

8.7.2 性能测试过程

Web应用性能测试的过程是一个重复循环、不断迭代的过程。首先，分析Web应用的真实运行情况，制定详细的测试计划，并构建一个尽可能真实的运行环境(少数性能测试就实施在真实的运行环境中)；然后，模拟多个用户并发对Web应用进行访问并生成测试结果；分析Web应用的性能并提交测试报告，最后根据测试报告分析系统瓶颈，优化被测Web应用，重新进行测试。

Web应用性能测试过程如图8.2所示。

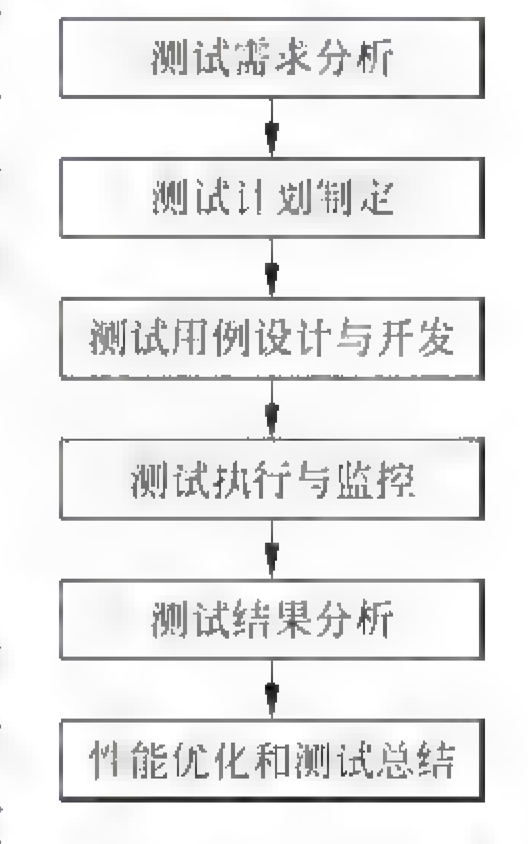


图8.2 Web应用性能测试的过程

1. 测试需求分析

测试需求分析是整个Web应用性能测试的基础。测试人员需要和Web开发人员与Web应用使用人员进行沟通，了解Web应用的体系结构和真实运行情况，收集用户对Web应用的性能需求。该阶段的主要任务是确定测试策略和测试范围。测试策略主要由用户关注度和Web应用业务特点决定。

2. 测试计划制定

在性能测试策略和测试范围确定后，需要制定性能测试计划。测试计划的内容主要包括测试范围、测试环境和测试方案简介。

测试过程必须以一个好的测试计划作为基础。尽管测试的每一个步骤都是独立的，但

是必定要有一个起到框架结构作用的测试计划。测试计划应该作为测试的起始步骤和重要环节。测试计划描述测试策略、资源和进度安排。计划内容包括：决定系统的负载量，决定开始测试的时间，决定测试过程是对硬件要求高还是对软件要求高，选择测试用例（确定测试内容），选择负载和压力测试工具，决定执行测试的人员，并确定测试要做到何种程度（测试的充分性）。性能测试计划不完全等同于传统软件测试计划，性能测试计划主要是分析处理在不同条件下的性能指示器的状态。一个测试计划应包括：产品基本情况调研、测试需求说明、测试策略和记录、测试资源配置、计划表、问题跟踪报告、测试计划的评审和结果分析等等。

3. 测试用例设计与开发

这一阶段的任务主要包括设计测试用例和开发测试脚本。开发测试脚本主要是指开发与用例相关的测试程序。比较常见的做法是先通过测试工具录制用户操作，然后进行代码修改和参数化工作。但并非所有的内容都能通过工具实现，对于有些工具做不到的内容，需要自行编写程序实现。

4. 测试执行与监控

这一阶段主要包括性能测试的实施和过程监控。测试实施是指通过测试工具或编写的代码来执行测试用例，也可以辅以真实用户来执行测试用例，具体工作有创建测试场景、执行测试场景、监控测试场景等。在执行过程中，通常会针对监控所得的运行情况对测试进行调整，如修改测试用例、调整测试范围甚至停止测试等。

5. 测试结果分析

根据执行测试所得到的测试数据、图表等来分析测试结果，为优化和调整系统提供依据。测试分析的对象包括应用程序、Web 服务器、Web 应用服务器、数据库服务器和硬件资源等。通过综合分析测试结果，准确定位整个 Web 应用的性能问题。

6. 性能优化和测试总结

根据测试结果及其分析得到的 Web 应用的性能评价和性能瓶颈，进行性能优化，可以针对 Web 服务器、网络及 Web 应用本身，如应用服务器调优。

8.7.3 性能测试内容

Web 应用性能测试可以划分为速度测试、负载测试、压力测试、并发测试、大量数据测试、配置测试和可靠性测试。

1. 速度测试

速度测试包括网络连接速度测试和业务处理速度测试。

用户连接到 Web 应用的速度根据上网方式和带宽的变化而变化。当下载一个程序时，用户可能需要等待比较长的时间，但如果仅仅只是访问一个页面，Web 应用响应时间太长（如超过 3 秒），就可能会导致用户没有耐心等待而离开，所以要对 Web 应用进行连接速度

测试。

业务处理速度测试目前主要是针对关键业务进行手工测速,可以在多次测试的基础上求平均值,可以与测试工具测得的响应时间等指标做对比分析。

2. 负载测试

负载测试主要是确定在用户可接受的响应时间内,系统能够承担的并发用户的数量。由于成千上万的用户可能在同一时刻访问同一个Web应用,为了保证系统的正常运行,必须测试在最重的工作负载下Web应用的运行情况。负载级别可以是某个时刻同时访问Web应用的用户数量,也可以是在线数据处理的数量。例如,Web应用能允许多少个用户同时在线,如果超过了这个数量,Web应用是否能承受得了,会出现什么现象,Web应用能否处理大量用户对同一个页面的请求,等等。

可以通过脚本来生成成千上万的"虚拟用户"同时访问Web应用并与Web应用进行交互。这些虚拟用户执行各种典型的任务,如浏览Web页面,购买商品,提交数据,搜索数据库,等等。在虚拟用户执行这些任务的同时,记录下服务器的响应时间。当测试执行完成以后,分析通过负载测试得到的数据,如在不同交互情况下的Web页面传送所需时间、Web页面传送出错信息等,经过一定的分析、计算、处理,得出Web应用能同时支持的用户数目、交互数目等,并尽可能找出多用户并发访问的瓶颈,最后以报告和图表的形式显示测试情况下Web应用的执行情况以及潜在问题存在的地方。

多用户访问的瓶颈可能发生在Web服务器、Web应用服务器或(和)数据库服务器上,要明确确定瓶颈所在并非易事。另外,要确定同时访问Web应用的用户数目并不容易。可以参考系统给出的参数结合实际情况进行估计,也可以选择有代表性的分布算法(如指数分布、常量分布、泊松分布等算法)来估计。

图8.3显示了经统计而得到的Web应用负载、Web应用响应时间与用户可能放弃的比例之间的关系。从图中可以看出并发用户数目越多,Web应用的响应时间越长,从而用户放弃执行该动作的可能性越大。因而必须从Web应用的实际负载能力出发,确定合适的并发用户数目,以先保证服务质量,然后在系统性能稳定的范围内,再考虑为更多的用户提供优质服务。

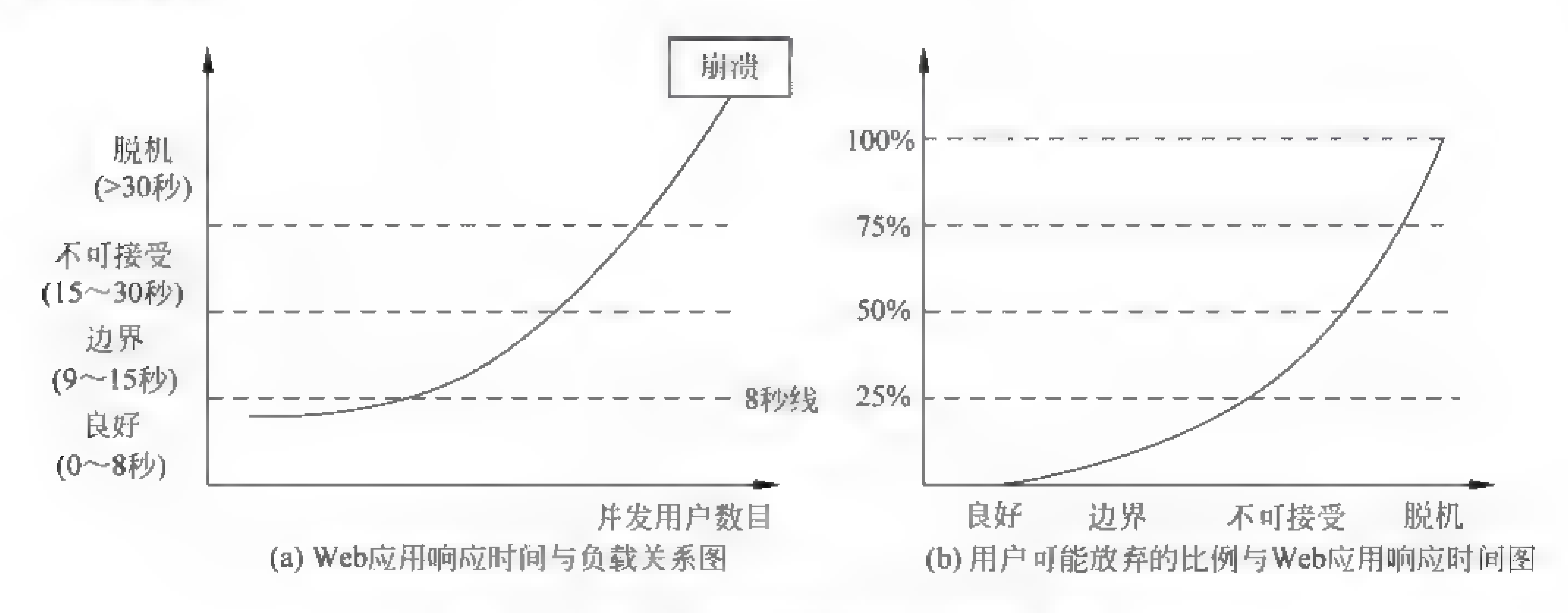


图8.3 多用户性能测度中部分数据关系图

负载测试应该安排在Web应用部署以后，在实际的网络环境中进行测试。因为一个企业内部员工，特别是项目组人员总是有限的，而一个Web应用能同时处理的请求数量将远远超出这个限度，所以，只有部署之后，接受负载测试，其结果才是正确可信的。

3. 压力测试

压力测试是负载测试的延续，通过对Web应用不断加压，来发现其在什么条件下变得不可承受，查出Web应用对异常情况的抵抗能力，找出性能瓶颈，从而获得系统能提供的最大服务级别的测试。压力测试关注Web应用能否在大量并发用户、传送大量数据和大业务量情况下发生性能变化，能否长时间运行，例如出现响应是否太慢、系统是否崩溃、能否恢复等情况。

在很多情况下，可能会有黑客或不法分子试图通过发送大量数据包来攻击服务器。出于安全的原因，测试人员应该知道当系统过载时，需要采取哪些措施，而不是简单地提升系统性能。

压力测试的主要手段是通过产生模拟业务对被测试系统进行加压，如以下做法。

(1) 当正常的用户点击率为"100/秒"时，运行点击率为"500/秒"的测试用例。

(2) 定量地增长数据输入率，检测对数据处理的反应能力。

(3) 运行需要最大存储空间(或其他资源)的测试用例。

(4) 运行可能导致操作系统崩溃或此版数据剧烈抖动的测试用例。

一个好的压力测试必须能够提供产生压力的手段，能够对后台系统进行监控，能够对压力数据进行分析并快速找出被测系统的瓶颈。现前流行的压力测试工具有LoadRunner、ACT、WAS和WebLoad等。

4. 并发测试

并发测试主要是指当测试多个用户同时访问同一个Web应用、同一个模块数据记录时是否存在线程同步问题、死锁或其他性能问题。其目的主要体现在：以真实的业务为依据，选择有代表性的、关键的业务操作设计测试案例，以评价系统的当前并发能力。

在实际测试工作中一般都借助工具来模拟并发访问，如LoadRunner。

5. 大数据量测试

大数据量测试主要测试运行数据量较大时或历史数据量较大时的性能情况，一般针对某些特殊的核心业务或一些日常比较常见的综合业务的测试。大数据量测试通常分为两种：①对某些存储、传输和统计查询等业务进行大数据量的测试；②与并发测试相结合的极限状态下的总和数据测试。主要测试运行数据量较大的或历史数据量较大时Web应用的性能情况。

由于大数据量测试一般在投产环境下进行，所以把它独立出来和疲劳强度测试一起进行，一般在测试后期进行。

大数据量测试分实时大数据量测试和极限状态下的测试。实时大数据量测试模拟用户工作时的实时大数据量，主要目的是测试用户较多或某些业务产生较大数据量时系统是否稳定；极限状态下的测试是指被测Web应用用了一段时间后，当数据已积累到一定程度

时，再查看系统是否正常。

6. 配置测试

配置测试指通过测试找到系统各项资源的最优分配原则，为系统调优提供依据。配置的可变性和不稳定性是 Web 工程面临挑战的重要因素。硬件、操作系统、浏览器、存储容量、网络通信速度和多种其他客户端因素对每个用户都是难以预测的。另外，某个用户的配置可能会有规律地改变（如操作系统升级、新的 ISP 和连接速度等），可能会导致客户端环境容易出错，这些错误既微妙又重要。如果两个用户不是在相同的客户端配置工作，一个用户对 Web 应用的印象以及 Web 应用的交互方式可能与另一个用户的体验有很大不同。

配置测试不是检查每种可能的客户端配置，而是测试一组可能的客户端和服务器端配置，以确保用户在所有配置中的体验是一样的，并且将特定于特殊配置的错误分离出来。例如，可以通过不停地调整 Oracle 的内存参数来进行测试，使之达到较好的性能。配置测试本质上是和其他性能测试一起进行的。

7. 可靠性测试

软件可靠性是指系统在规定条件下，在规定时间内，软件无失效工作的概率。概率受输入和使用以及其存在故障的影响，系统输入将确定是否会遇到存在的故障。而可靠性测试是指给系统增加一定业务压力的情况下，让系统运行一段时间，以此来检测系统是否稳定。

Web 应用的可靠性测试的目的如下。

① 通过在有使用代表性的环境中执行 Web 应用，以证实 Web 应用需求是否正确实现。

② 数据的准确性关系到 Web 应用可靠性评估的准确度，通过数据采集、模型选择、模型拟合以及系统可靠性评估 4 个步骤，为进行 Web 应用可靠性估计采集准确的数据。

③ 找出所有对 Web 应用可靠性影响较大的错误。

8.7.4　性能测试方法

Web 应用的性能测试需要有良好的测试方法来保证。执行 Web 应用性能测试的方法有许多种，它们各有优势，其中最具有代表性的主要方法有：虚拟用户方法、WUS（Website Usage Signature，站点使用签名）方法和 SPE（Software Performance Engineering，软件性能工程）方法。

1. 虚拟用户方法

虚拟用户方法通过模拟真实用户的行为来对待测 Web 应用施加预期工作负载，以测量待测系统的性能，如事务的响应时间、服务器的吞吐量等。它以真实用户平时工作环境下的“事务处理”（用户为完成一个商业业务而执行的一系列操作）作为负载的基本组成单位，用“虚拟用户”（模拟用户行为的测试脚本）来模拟真实用户。工作负载的信息（如并发虚拟用户数、商务处理的执行频率等）通过人工收集和分析系统的特征来获得。

支持该方法的性能测试工具可用较少的硬件资源模拟出成百上千虚拟用户同时访问，并可模拟来自不同的地址、不同浏览器类型以及不同网络连接方式的请求，同时可实时监控

相关性能指标，帮助测试人员分析测试结果。该方法易于实现和便于操作，已经为绝大多数主流的Web应用性能测试工具所采用，在实践中被证明是相当有效的。

这种方法需要测试人员有相当高的专业技能，需要根据预期工作负载设计准确的测试场景，以较好地模拟工作负载进行测试。

2. WUS方法

基于WUS的概念来设计测试场景，强调建立真实的负载。WUS是为了衡量测试负载和真实负载之间的接近程度，是一系列能全面刻画负载的参数和测量指标的集合，包括每小时浏览的页面、平均访问持续时间、每次访问平均浏览的页面以及页面请求分布等。这些参数可以从日志文件中得到，即需要Web应用的运行日志，也就是说Web应用正常运作一段时间后才能进行测试。参数同时也包括一些影响负载的客户端变量，如用户对站点的熟悉程度、对延迟的忍耐程度和客户端连接速度等。

此外，采用类似的Web应用的日志信息来模拟待测的Web应用工作负载方法，理论上也是可行的，只不过这个条件在现实测试环境中非常苛刻，基本上难以实现。

3. SPE方法

SPE方法是一种控制性能的综合方法，是由美国学者Connie U Smith等人提出的面向软件开发的一种性能测试方法。它是一种系统的、定量的方法，用于构建能够符合预期性能目标的软件系统。

它基于一般软件基础，通过模型来预测和评估软件功能、硬件规模、质量结果和资源需求之间的平衡点。它的核心内容是建模，以系统模型为基础，通过建立抽象而准确的软件处理过程模型，判定目标软件是否能够满足性能目标。随着软件开发过程的推进，对模型进行持续的细化，使其更加精确地反映正在开发的软件的系统特征，并重新进行性能测试和评估。

SPE方法是白盒方式的Web应用性能测试论的代表，它把性能测试纳入了工程的范畴，侧重于软件开发时就进行相应的系统性能检测和评估，避免在软件系统开发完成后因性能问题重新设计和开发。

8.8 安全性测试

Web应用运行在Internet上，其安全性一直广受关注，Web应用安全测试则是一个复杂的主题。Web应用安全性测试是对整个Web应用的安全防卫措施的有效性进行测试，以揭露安全机制中的漏洞。通常的安全机制包括信息访问控制、用户身份校验以及对机密信息进行加密等，同时要能根据用户的访问情况判断出该用户是正常的用户还是蓄意的破坏者。

Web应用的安全性测试主要包括数据加密测试、用户身份验证测试、日志文件测试、Session测试、备份与恢复测试等内容。

1. 数据加密测试

数据加密测试是测试 Web 应用使用的关键数据是否经过了加密，所选择的加密算法是否合适等内容。

2. 用户身份验证测试

Web 应用中用户身份验证直接关系到用户权利和隐私的安全。用户身份验证测试主要检查无效的用户名和密码能否登录，密码是否对大小写敏感，是否有验证次数的限制，是否存在不验证而直接进入 Web 应用的问题，是否存在不登录就可查看非会员页面和权限问题。

3. 日志文件测试

日志文件测试主要检查 Web 运行的相关访问和状态信息是否写进了日志文件，是否可追踪，等等。此外，需测试 Web 应用执行过程中所产生的错误是否作为日志保留下来，以供技术人员分析该错误是由系统实现漏洞引起的还是由于黑客攻击引起的。

日志测试主要的测试内容如下。

(1) 日志是否记录所有的事务处理。

(2) CPU 的占有率是否很高。

(3) 是否有例外的进程占用。

(4) 是否记录失败的注册企图。

(5) 是否记录被盗信用卡的使用。

(6) 是否在每次事务完成时都进行保存。

(7) 是否记录 IP 地址。

(8) 是否记录用户名等。

4. Session 测试

Session 测试主要检查 Web 应用是否有超时的限制，也就是检查用户登录到 Web 应用后在一定时间内没有点击任何页面，是否需要重新登录才能正常使用，检查超时后能否自动退出，退出之后，浏览器回退按钮是否可以回到登录页面，等等。

5. 备份与恢复测试

根据 Web 应用对安全性的要求可以采用多种备份与恢复手段，如数据库增量备份、数据库完全备份、数据库差量备份和系统完全备份等。出于更高的安全性要求，某些实时系统经常会采用双机热备或多机热备。除了对这些备份与恢复方式进行验证测试以外，还要评估这种备份与恢复方式是否满足 Web 应用的安全性需求。

6. 访问控制策略测试

访问控制策略测试主要检查管理接口是否只有授权的管理员才允许进行访问，是否有完善的访问控制策略文档，文档中是否精确定义了每类用户可以访问的功能和内容，执行访

问控制策略的代码结构是否严谨，等等。

7. 安全漏洞测试

为了避免或减少Web应用被攻击的可能性，需要测试并发现其中隐藏的跨站脚本、命令注入、SQL注入等安全漏洞，这类测试可借助一些安全扫描工具来实现。

8. TCP端口测试

开放不必要的端口，会给Web应用带来安全隐患。所以在部署Web应用之前，要用端口扫描软件对部署环境进行TCP端口测试，确保只开启了必要的端口。

9. 服务器端脚本漏洞检查

服务器端的脚本漏洞检查是测试存在于服务器端的脚本是否存在安全漏洞，没有经过授权能不能在服务器端放置和编辑脚本。

10. 防火墙测试

防火墙测试是对防火墙功能和设置进行测试，以判断是否满足Web应用的安全需求。

8.9 接口测试

Web应用不是孤立存在的，它可能会与外部服务器通信，请求数据、验证数据或提交订单，所以需要进行接口测试。Web应用接口测试主要包含有服务器接口测试、外部接口测试、错误处理测试三个方面的内容。

1. 服务器接口测试

首先需要测试的接口是浏览器与服务器的接口。接口分为请求和返回接口两大类。要测试请求接口是否正确，测试人员可以提交事务，然后查看服务器记录，并验证在浏览器上看到的正好是服务器上发生的。测试人员还可以查询数据库，确认事务数据已正确保存。检查返回的接口，可以查看系统在提交后回显是否正确。有的系统也许看不到返回的信息，就可以到服务器上查看返回的信息是否符合接口要求。

2. 外部接口测试

有些Web应用有外部接口，要测试Web应用与这些接口之间的交互。例如，网上商店可能要实时验证信用卡数据以减少欺诈行为的发生。测试时，要使用Web接口发送一些事务数据，分别对有效信用卡、无效信用卡和被盗信用卡进行验证。通常，测试人员需要确认软件能够处理外部服务器返回的所有可能的消息。

3. 错误处理测试

最容易被测试人员忽略的地方是接口错误处理。通常人们试图确认Web应用能够处理所有错误，但却无法预期系统所有可能的错误。尝试在处理过程中中断事务，观察处理情

况；尝试中断用户到服务器的网络连接，尝试中断 Web 服务器到信用卡验证服务器的连接。在这些情况下，系统能否正确处理这些错误，是否已对信用卡进行了收费处理，如果用户自己中断事务处理，在订单已保存而用户没有返回 Web 应用确认时，就需要外部确认。

8.10 Web 服务测试

Web 服务在面向服务计算中得到了越来越广泛的使用，它通过在运行时动态发现和绑定服务来进行服务集成。这给 Web 服务测试，尤其是服务之间的交互测试，带来巨大的挑战。

1. Web 服务测试特性

Web 服务所固有的新特征给测试带来了一系列的挑战，主要表现在以下几个方面。

(1) Web 服务是被不为人所知的开发人员开发，基于一系列由 XML 构成的服务规范，对合作者而言，源代码不可见，对服务的测试必须基于标准规范，使得对其进行质量测试面临很大挑战。

(2) Web 服务是在一种不可预知的环境中运行，如访问的用户类型、并发用户数量、Web 服务调用的装载模式和访问方式等，再加上其分布式的特性，都增加了 Web 服务测试的挑战。

(3) Web 服务的发布、绑定、调用和集成都是在一种动态的环境中进行的，其过程的不确定性和不可见性增加了测试的挑战性。

(4) Web 服务间互操作和消息传递在分布式环境中进行，对 Web 服务的安全性和消息的传送量和响应时间也提出了更高的要求。

(5) Web 服务以 WSDL 文档进行发布，增加了 Web 服务的安全隐患，提高了被攻击的机会。此外，对于所调用的分散、异构的外部 Web 服务的安全性的管理更为困难。如何提高 Web 服务的安全性也是测试者要考虑的重要问题。

(6) Web 服务通常涉及服务提供者、服务代理和服务请求者三种角色，这三种角色都需要参与到测试的不同阶段。其分布式协作的特征使得测试的组织缺陷管理、结果评估等活动都更加困难。

与传统软件测试相比，由于 Web 服务自身的特性，使其测试也具有不同特性，如表 8.2 所示。

表 8.2 Web 服务测试特性

测试因素	Web 服务测试
测试参与者	需要服务的提供者、代理和请求者共同参与到测试过程中
测试分布	分布、远程、多阶段
测试模型	不拥有服务代码(服务开发者除外)，有限的测试模型(可控制性和可观测性低)
测试覆盖	除服务开发者拥有传统的覆盖范围外，服务提供者、服务集成者、服务请求者和服务代理可能只有黑盒覆盖范围
测试执行	在线、及时测试
测试客户端	需要构建测试客户端调用被测服务
测试预测	Web 服务动态绑定，很难预测其实际运行情况，难以生成测试预测
回归测试	在线、基于运行时收集数据的测试；除服务的开发者和提供者外，服务请求者、服务集成者和服务代理并不掌握服务的演化情况，回归测试需要额外的信息支持

2. Web服务测试内容

根据Web服务架构和业务模型，Web服务测试可分为三个层次：基础设施测试、Web服务独立测试以及Web服务集成测试。同时，测试组织和管理是Web服务测试的三个层次中都需解决的问题，对测试的系统性和有效性至关重要。

1）Web服务基础设施的验证与确认

虽然标准化和开放性是Web服务的主要宗旨，但与传统的应用系统相比，Web服务中间件的稳定性和可靠性相对薄弱。一方面，是由于Web服务的整个技术架构还尚未成为成熟的产业标准。在W3C联盟以及相关研究机构的大力支持下，标准规范体系还处于不断发展和完善的过程中，不同的标准之间的概念模型和表示体系可能互不兼容，并进一步导致相关应用程序的不兼容；另一方面，Web服务规范体系采用XML作为基本的编码语言。由于XML技术的简单、灵活、可伸缩、易定制，基于XML数据描述和XML Schema数据建模定义的Web服务的协议栈具有标准化和开放性的特点。同时，XML为描述性语言，其通用性和灵活性为协议的可证明性提出了挑战。当Web服务用于高可信性或实时要求较高的应用中，需要更严格的验证方法。例如，SOAP RPC是Web服务的基础和核心协议，由于编码格式的不同，SOAP RPC与传统的DCE RPC及ONC RPC采用了完全不同的消息映射和处理机制。SOAP协议还处于完善过程中，对于SOAP协议本身的验证还有待于进一步加强。

2）Web服务独立测试

Web服务首先作为独立的功能结点发布，再通过工作流定义和解析动态集成为完整的业务流程。Web服务独立测试就是从以下三个方面保证各服务结点的质量。①服务的实现应在功能、性能等各方面与发布的服务描述相一致。为验证一致性，除服务提供者外，服务代理及用户都应能在一定的安全约束下，远程测试该服务。②由于服务发布的开放性，对于每一个服务请求，可能存在多个满足需求的服务描述，服务代理应能根据一定的度量和评价标准，对多个服务进行测试、比较和评估，并依照需求的满足程度排序。③在服务实现的演化过程中，应建立一定的机制来支持对不同版本的跟踪及回归测试。

3）Web服务集成测试

通过对服务流描述的解析，Web服务可以动态地集成。Web服务集成的描述、解析和执行将是Web服务区别于其他分布式计算技术的一个主要特征。目前已经提出了多种Web服务描述语言，如IBM的WSFL，微软的XLANG以及IBM的BPEL4WS。集成的Web服务测试就是在服务流描述执行前，通过静态验证以及动态模拟的方法，确认服务描述能够正确地描述业务需求，能够由服务中介正确地解析，并能由所有服务结点正确地执行。

8.11 测试工具

由于Web页面数目巨大且变动频繁，如果单靠手工对其进行测试显然是无法进行的，必须有测试工具的参与，而测试工具的应用已经是Web应用测试的普遍趋势。目前用于Web应用的自动化测试工具众多，这些测试工具一般可分为白盒测试工具、黑盒测试工具、性能测试工具和安全测试工具等。自动化测试工具的使用可大大提高Web应用测试的效率。

常用的测试工具应该要具以下功能。

① 测试计划与管理。这些工具使得对于测试用例、测试数据、合适的测试用例的选择、测试结果的收集、Bug 追踪的管理变得容易。

② 测试用例的设计。对于 Web 应用开发人员而言，这些工具支持设计测试用例，可以根据需求定义产生测试用例或产生测试数据。

③ 静态和动态分析。这些工具可以分析 Web 应用。

④ 自动运行测试。

⑤ 系统监控。这些工具支持系统监控，例如，可以捕获系统的性能(包括内存消耗、数据库访问等)。

总的来说，测试工具的应用可以提高测试的质量和效率，减少测试人员的工作量。但是在选择测试工具和使用测试工具时，也应该看到，在测试过程中，并不是所有的测试工具都适合使用。同时，有了测试工具、会使用测试工具并不等于测试工具真正能在测试中发挥作用，这些都是 Web 应用测试中需要注意的问题。测试工具对于保障 Web 应用的功能、性能起到了重要的作用。以下详细介绍了常用的几种测试工具，Web 应用测试人员可根据项目的实际需要，结合各测试工具的特性，选择适合自身的测试工具，从而提高测试效率。

1. 功能测试工具

功能测试工具一般采用录制-回放(Record & Replay)的方式来代替重复的、繁重的手工测试操作，这种方法在回归测试中尤其有效。常见的 Web 功能测试工具包括 WinRunner、Rational Robot 和 Selenium 等。

WinRunner 是 Mercury Interactive 公司的一种企业级的功能测试工具，通过自动录制、检测和回放用户的应用操作，能够有效地帮助测试人员对复杂的企业级应用的不同发布版进行测试。IBM Rational Robot 可以对使用各种集成开发环境和语言建立的软件应用程序，创建、修改并执行自动化的功能测试、分布式功能测试、回归测试和集成测试。Selenium 是一个用于 Web 应用程序测试的工具，它直接运行在浏览器中，能测试与应用浏览器的兼容性以及系统的功能。

2. 性能测试工具

性能测试工具能够模拟很多的用户同时访问 Web 应用，以便进行请求数据、提交事务以及其他的电子商务活动，同时，虚拟负载也能够模拟各种版本 Web 浏览器和网络带宽。当模拟负载应用于服务器时，性能数据被收集并可以生成指定格式的测试报告，为进一步的分析做准备。常见的性能测试工具有 JMeter、LoadRunner 和 WAS 等。

JMeter 是 Apache 组织的开放源代码项目，可以用于测试静态或者动态资源的性能。LoadRunner 是一种预测系统行为和性能的负载测试工具，通过模拟上千万用户实施并发负载及实时性能监测的方式来确认和查找问题，能够对整个企业架构进行测试。WAS (Microsoft Web Application Stress Tool)是由微软的 Web 应用测试人员所开发，专门用来进行实际 Web 应用压力测试的一套工具。

3. 安全性测试工具

安全测试工具采用各种静态或者动态的分析技术，查找系统中隐藏的各种安全漏洞。

常见的Web安全测试工具包括IBM Rational AppScan、HP WebInspect和Acunetix Web Vulnerability Scanner(WVS)等。

AppScan扫描Web应用的基础架构，进行安全漏洞测试并提供可行的报告和建议。HP WebInspect是一款易用、可扩展、精确的Web应用安全评估软件，可协助发现Web应用和服务中存在的高风险漏洞。WVS是一个自动化的Web应用安全测试工具，它通过引入高级的启发式发现技术，可以通过检查SQL注入攻击漏洞、跨站点脚本攻击漏洞等来审核Web应用。

4. Web服务测试工具

在进行Web服务测试的时候，多数情况下必须花费大量精力来编写和调试自己的客户端，并且往往缺乏通用性。因此必须使用Web服务测试工具根据WSDL的URL地址自动产生客户端进行Web服务测试。目前市场上有部分的Web服务测试工具，其中比较常见的有Parasoft的SOAtest、SOAP UI以及IBM Rational Tester。这些测试工具都提供图形化的界面以方便用户使用，允许用户导入WSDL文档，手动创建和编辑测试用例，运行测试用例和检查结果。其中的一些工具还能够根据WSDL文档所包含的信息，用随机、等价类划分、边界值分析等方法自动化地产生一些测试数据。

Parasoft SOAtest是一款自动化测试Web服务的产品，支持WSDL有效性验证，客户/服务端单元及功能测试、性能测试等。SOAP UI是一款基于Java的开源工具，它主要包含的功能体现在Web服务功能测试、负载测试、断言测试、批量测试、模拟测试等，并且可以用Groovy脚本语言将测试集组合起来一起测试。IBM Rational Tester是自动化SOA和Web服务测试工具，能够自动创建、执行和分析功能和回归测试。

8.12 总结与展望

Web应用测试的目标是对系统的质量维度进行检查，找出隐藏的错误或缺陷。但是，Web应用测试由于其质量需求范围更大，需要关注短的响应时间、易学易用、愉悦的观感、高安全、浏览器的兼容性、内容的更新等，所以它与传统的软件测试又有所区别。在传统软件测试的基础上，主要针对的是Web应用的内容、功能、结构、可用性、导航性、性能、兼容性、互操作性和安全性等质量属性。

Web应用的持续改进使测试面临新的挑战。测试应该在多个开发周期之间可重用，应该能够监控Web应用的运行；测试还应该具有可适应性，使有关可重用性和可变更性的重要的质量需求为测试本身而提升。因此，自动化测试在大型Web应用中因其可重用性而会获得更大的市场。敏捷测试方法已经将自动化测试作为必要条件纳入其中。此外，随着SOA的发展，越来越多的Web应用采用Web服务技术，遗留的Web应用也开始采用Web服务进行包装整合，这将使Web服务的测试成为Web应用测试的一个重要方面。

Web应用的动态性越来越强，许多页面内容都是动态生成的，这就对测试中动态分析技术提出了更高的要求，需要提供更为有效的动态分析手段。目前的自动化测试技术和工具还存在亟待解决的问题，最为典型的就是测试数据的自动生成问题，所以自动化测试技术还将是Web测试的一个重要研究方向。

第 9 章 Web应用的运行和维护

Web 应用部署完成并上线(启动)运行后,就进入运行和维护阶段。Web 应用维护是指 Internet 运营体系中一切与 Web 应用后期运作有关的维护工作,它是 Web 应用生命周期的最后阶段,关系到 Web 应用生存的重要标准。与其他媒体一样,Web 应用也是一个媒体,需要经常性的更新维护才能起到既定的效果,因此 Web 应用运营维护的好坏在很大程度上会直接影响到用户的满意程度。只有运营维护良好的 Web 应用,才能在瞬息万变的信息社会中抓住更多的网络商机,才能获得更多的用户的喜欢,最大限度地获得用户访问量。

传统应用软件的运行和维护阶段划分得非常清楚,而 Web 应用由于其自身的特性,很少有 Web 应用能在开发构建真正完成之后才上线运行,很大一部分开发工作是在运行和维护阶段完成。因而,对 Web 应用的运行与维护是一个持续、漫长、艰辛的过程,需要规划 Web 应用维护的方针和策略。Web 应用上线运行后如何推广以被搜索引擎搜索进而使用户了解,以及随着 Web 应用的需求的增加和变更,Web 应用内容需要被更新,甚至可能需要进行重新设计来满足新的需求。同时,对 Web 应用潜在的安全风险、系统性能和使用情况进行分析和评估也非常重要,它直接影响着用户对 Web 应用的满意程度。通过 Web 应用的维护,解决 Web 应用中发现的诸多问题,进而提高 Web 应用的质量。

9.1 Web 应用运行和维护的挑战

任何一个系统都不是一开始就很完美,总是经过多重的开发、运行、再开发、再运行的循环不断提升的。

Web 应用作为一种新兴的传播媒介与共享资源,它并非一次性投资,一般是长期的投入,在构建完成启动运行后需要推广、长期更新和维护,而这一点又往往在项目前期被严重低估。在 Web 应用运行的过程中,可能遇到很多问题,如遇到特殊情况下运行不正常,Web 页面不符合用户的操作习惯,Web 应用突然无法打开,Web 应用加载速度变慢,服务器受到了非法攻击,并发访问量超过了服务器的可承受能力,甚至内容陈旧,等等,这些都可能导致 Web 应用不为用户接受或者不可访问等。

运行 Web 应用可以及时地发现 Web 应用中可能存在的潜在问题,便于以后的维护工作。针对这些暴露出的问题,充分考虑企业的商业动态和发展方向,结合现有的 Web 应用

规划结构，设计解决方案，并迅速做出相应的改进，解决这些隐患，从而提高Web应用的可用性。所以Web应用维护工作是个要求知识全面、技术含量高和长期性的工作。

Web应用的运行与维护面临着巨大挑战，主要表现在以下几个方面的内容。

1）用户群不确定

相比传统应用软件开发而言，Web应用的部署、运行和维护给开发人员和运行人员造成了更多的交付挑战，但在计划阶段却经常会被低估。主要原因是Web应用运行在Internet上供全球用户访问，而传统的应用软件却有着特定的用户群。Web应用的用户群涉及的物理范围十分宽广，有时会遍及全球。很多支持跨国业务的Web应用往往需要考虑多语言的支持问题，而且还要考虑到不同地域不同国度的文化和使用习惯的特点。用户群和用户特性的多种多样及不断变化，会造成Web应用需求的不确定。

2）开发周期短

Web应用的部署阶段的重要特性就是它的实时性和全球性。部署Web应用之后，关键的焦点就会变成分析Web应用的适应性，分析市场需求和用户兴趣与Web应用的结合度。对于一个Web应用而言，经常需要达到24小时×7天的高可用性以及超预计的并发用户访问和无限制数量的用户访问能力，这就需要尽最大努力考虑节省在软硬件设备上和网络资源方面的花费。而面对Web应用开发周期短、更新周期更短的情况，也必须考虑在预算计划范围内。

3）相关工作流的互相影响

另一个重要的影响Web应用成功与否的因素是将Web应用嵌入到相关的公司或组织的工作流中。在很多情况下，对与Web应用直接或间接相关的工作流的重定义是完全有必要的。这通常也是增加产量、减少周期，从而提高质量保证的唯一途径。例如，对于商业类Web应用，工作人员可以设定Web应用是通过电话还是通过电子邮件连接到帮助台。电子邮件连接的咨询回答通常是免费的，但交互的实时性不高。而对于电话这种连接方式而言，实时性较高，但需要考虑通信成本，电话的收费方式通常是以增值的方式进行的，例如，以通话时间作为收费标准。对于电子商务应用而言，Web应用需要和其他业务系统充分交互和沟通，保证整体工作流程的顺畅和高效。比如，旅游信息平台可以为用户提供信息和预定支持，与呼叫中心平台之间通过回调函数实现交互，共同辅助网上用户完成电子预定过程的工作流，帮助他们找到匹配的旅游套餐。

Web应用维护是一项专业性较强的工作，其维护的内容也非常之多，例如，将Web应用通过搜索引擎等方式进行推广；服务器及相关软硬件的维护，包括提高硬件配置等，对可能出现的问题进行评估，制定防护策略；数据库维护，有效地利用数据；Web应用内容的更新、调整、增加、删除等；Web应用的页面风格、版式等的更新与改善；制定相关Web应用维护的规定，将Web应用维护制度化、规范化；安全防护，发现Web应用的安全威胁，并及时消除威胁；等等。这些维护都需要懂得相关知识的专业人士来完成。做好Web应用的维护需要统筹兼顾、集思广益，最大限度地满足用户的需求。只有做好了这一点，才能发挥Web应用应有的功效。

以下分别从推广营销、内容维护和Web使用挖掘几个方面分析Web应用的维护。

9.2 推广营销

让更多的人知道、了解和访问Web应用，是Web应用开发人员最大的期望。Web应用构建的真正目的是要使这个Web应用取得效果，能够吸引尽可能多的访问人员。Web应用的推广营销指采用多种方法，让更多的用户了解并访问Web应用，以提高知名度和影响力，尽最大可能提高访问量，吸引用户和创造商机，最终达到Web应用构建的宗旨。

推广营销是个系统工程，从Web应用的构建时期开始规划，妥善处理Web应用推广所涉及的关联问题。在策划与构建阶段就需要充分考虑，对Web应用推广需求的技术和预算支持、推广的时机、推广的手段和方案等。不同的推广方法对同样的Web应用可能会产生明显不同的效果，因此需要考虑推广效果的影响因素，并选择适当的推广方式。

Web应用的推广一般采用网络广告、搜索引擎营销、传统广告等策略。本节重点分析一些基于网络的常用推广营销方式。

9.2.1 网络广告

网络广告(Webvertising)是常用的基于Web的网络营销策略之一，对Web应用推广有明显功效。它是一种以消费者为导向、个性化的广告形式，消费者可以根据自己的个性特点和喜好，选择是否接收以及接收哪些广告信息。企业应在自己的Web应用的每张Web页面上设置横幅广告，强调对自身特点的选择，加深用户印象。同时要尽量到别的Web应用上放置自己的广告，以达到宣传效果。为了获得最好的营销效果，除了免费的网络广告交换外，通常还要在其他访问量高的、有影响力的Web应用或者潜在顾客集中的Web应用购买广告空间。将网络广告用于Web应用推广，具有可选择网络媒体范围广、形式多样、适用性强、投放及时等优点，适合于Web应用发布初期及运行期的任何阶段。

网络广告作为一种强有力的推广手段，它的传播冲破了时间和空间的限制，只需要网络就可以把广告信息不间断地传递到世界的每一个角落。常见的网络广告类型有横幅式广告、关键词广告、赞助商广告、邮件列表广告、电子邮件式广告、插页式广告、互动游戏式广告、网上分类广告、网络短片广告等类型。常见的网络广告方式有时事通讯和网络联盟。

1. 时事通讯

企业进行必要的搜索和分类查询找到和企业有关的Web应用之后，企业可以通过电子邮件、QQ等应用软件群发给其联系人发送时事通讯(Newsletter)，用非常友好、诚恳的语气邀请对方服务企业的Web应用，并向其介绍企业的Web应用的内容如何与他们有关，有哪些更新，以引起他们的兴趣。通过电子邮件发送时事通讯是Web应用推广的重要方法之一，不仅对已有客户提供如产品或技术等方面更新信息，还吸引新的客户关注这些新产品和新技术，但要注意其内容的目标明确、简洁、及时、定期发送，以及取消订阅选项，等等，而且广告不可过度。

2. 网络联盟

商务类Web应用可以与门户类Web应用结盟，利用门户类Web应用的品牌效应和大

量的访问流量，可以获得大量的眼球注意，从而扩大商务类Web应用的知名度与访问量。如中国电子商务站点8848与中文门户搜狐(sohu.com)曾于2000年1月17日在北京宣布结成联盟站点，国外知名的Web应用亚马逊书店与美国在线也有很好的战略伙伴关系。在网址导航类站点添加链接，例如http://www.hao123.com，利用这些站点提供的导航服务，从而带来大量的流量。另外，Web应用经常与其他Web应用之间建立联盟，其好处在于企业之间整合资源合作推广，提升双方的访问量，而且企业还可以相互借鉴，以增加提高访问率的经验，寻求解决某些问题的方法。

1）资源合作推广

通过Web交换链接、交换广告、内容合作、用户资源合作等方式，在具有相关目标Web应用之间达到相互推广。其中最常用的资源合作推广方式为Web应用交换链接策略，利用合作伙伴之间访问量进行资源合作推广。

链接是决定Web页面权重最重要的因素。交换链接也称互惠链接、互换链接、友情链接，是开展网上营销的最经济、最便利的手段之一。Web应用之间通过交换图片或文字链接，使本Web应用访问者很容易到达另一个Web应用(对新闻站点尤为重要)，这样可以直接提高访问量，扩大知名度，实现信息互通、资源共享。

交换链接的作用主要表现在获得更多访问量，增加用户浏览时的印象，在搜索引擎排名中增加优势，通过合作Web应用的推荐增加访问人员的可信度，等等。交换链接还有比直接链接效果更深一层的意义，链接的访问量高的Web应用比链接的访问量低的Web应用更有优势。一般而言，每个Web应用都倾向于链接一些流量比自己高的、知名度更高的Web应用。

交换链接要注意以下几个方面的要点。

(1) 互相交换的Web应用应属于同一领域或相关领域，内容具有相关性。如汽车销售类与汽车维修类相关性很强，而与服装设计几乎无任何相关。

(2) 链接在页面的具体位置能极大地影响被用户点击的机会。交换链接置于Web应用首页的显要位置，被点击的概率就大，而位于内页等次要位置，被点击的概率相对较低。交换链接的意义受到Web应用访问量的影响。

(3) 合作双方要本着互赢的目的，公平的商业条件，增加吸引力或访问量，都是合作双方达到长期合作的重要基础。

进行交换链接时，也会存在一些误区，比如Web应用页面的评价越高越好，链接数量越多越好。其实不然，如果是不同类型的Web应用，页面评价值即使高也很难达到应有的效果。链接数量也是如此，如果很多链接都是垃圾链接或是由"链接农场"提供的无用链接，那么这些链接对自身的Web应用的排名是百害而无一利。

2）在B2B上发布信息或进行企业注册

B2B(Business to Business，商业对商业)是借助网络的便利条件，在买方和卖方之间搭起的一座沟通的桥梁，买卖双方可以同时在上面发布和查找供求信息。具有代表性的B2B应用有阿里巴巴(Alibaba)、美商网(MeetChina.com)等。

9.2.2 搜索引擎营销

搜索引擎(Search Engine)并非自动或立即就能找到新上线的Web应用，它是根据一定

的策略，运用特定的网页抓取程序（通常通过 Spider、Robot 或 Crawler 等）搜集互联网上的信息，对信息进行组织和处理后，为用户提供检索服务的系统。搜索结果中既包括组织好的搜索结果，又包括付费广告（如 Sponsored Links）。其主要工作包括页面收录、页面分析、页面排序及查询，工作原理如图 9.1 所示。

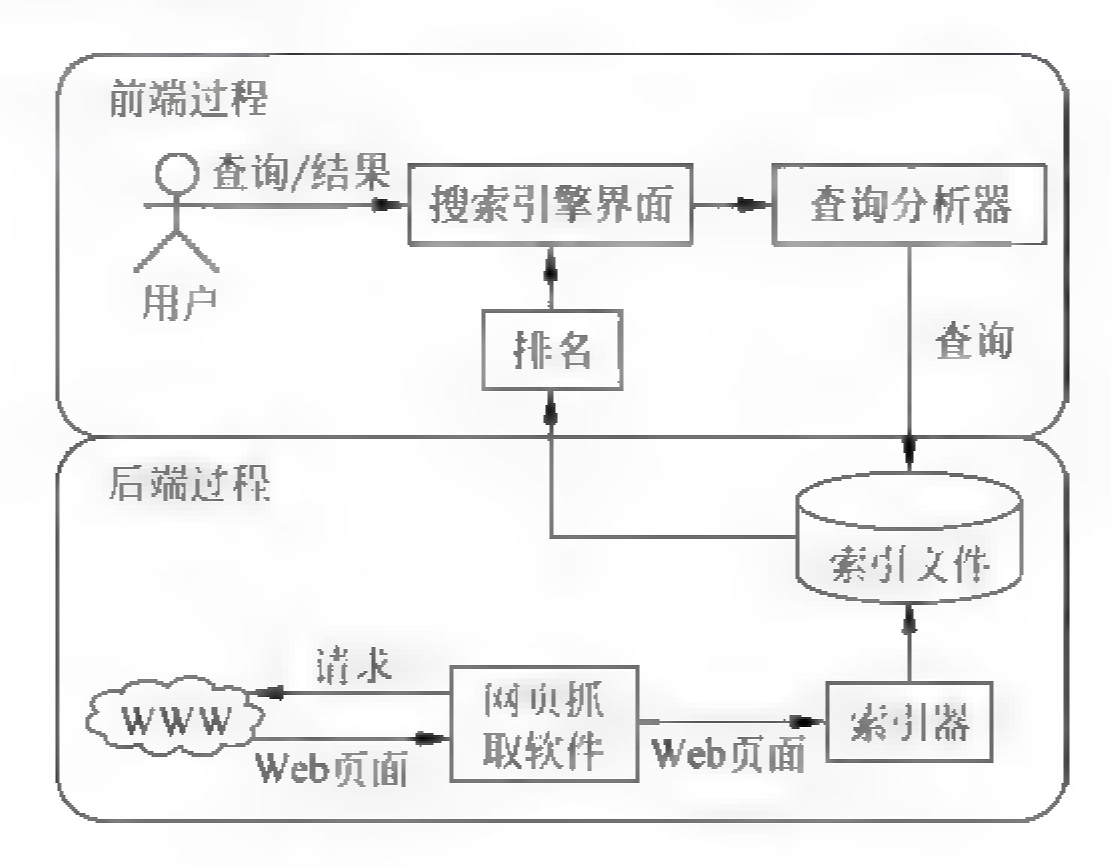


图 9.1 搜索引擎工作原理

页面收录是指搜索引擎在互联网中进行网络信息搜集，然后将搜集到的信息传输并存放到本地。页面分析指搜索引擎对收录的页面将进行一系列的分析、处理，如过滤标签提取页面正文信息，对正文信息进行切词处理，建立关键字与页面间的索引等，建立索引数据库或目录指南。并通过算法，给每个收录的 Web 页面评价并决定其最终排名，为用户的查询做好准备。用户向搜索引擎提交关键字查询信息后，通常会返回多个结果页面，决定页面排序的主要因素包括页面相关性和链接权重。

搜索引擎营销是根据用户使用搜索引擎的方式，利用用户检索信息的机会将营销信息传递给目标用户。搜索引擎营销是网络营销方法体系中的重要内容，以较高的投入产出比、目标针对性强等优势，成为 Web 应用推广的首要方法。

搜索引擎营销主要有以下几种模式：搜索引擎登录与排名、SEO（Search Engine Optimization，搜索引擎优化）、关键词广告、域名策略等。

1. 搜索引擎登录与排名

除非一个 Web 应用有非常非常多的页面链接到它，否则需要登录（Submission）搜索引擎，而非坐等搜索程序来查找。搜索引擎登录有免费和收费两种方式。免费登录需要几周的时间才能真正登录成功；收费方式可以大大加快登录速度，不过需要周期性地再提交，如每个月一次。

登录搜索引擎的方式比较简单，根据搜索引擎的提示逐步填写。如果搜索引擎收录 Web 应用需要人工审核，当搜索引擎的管理人员收到用户提交的信息后一般会通过对 Web 应用的访问，判断用户所提交的内容是否属实，以及用户所选择的类别是否合理，以决定是否收录该站点。新收录的 Web 应用一般可以在几天到几个星期之后，搜索引擎数据库更新时显示出来。如果 Web 应用因为提交时尚未构建完成或者质量不高被拒绝登录，那么需要经过改进后过一段时间再提交。

绝大多数用户只查看搜索结果中前10到20条，因此，Web应用排名是一个重要的营销特性。因为无法确定搜索引擎是如何计算搜索结果的排名，计算策略经常变化，而且各Web应用相互竞争获得好排名，所以也就无法保证或强制Web应用被列在比较靠前的位置，因而可以通过一些SEO和竞价排名等方式提高排名。

竞价排名采用竞价方式来提高Web应用在有关竞价关键词搜索结果中的排名位置，金钱是决定排名的重要因素。它们的搜索结果实质上是一种出现在搜索结果中的关键词广告，广告主根据用户的点击付费，每个点击的价格由排名位置决定，排名越高，广告主支付给搜索引擎的钱也越多。关键词广告是收费搜索引擎营销的主要模式之一，也是目前搜索引擎营销方法中发展最快的模式。它与一般网络广告的不同之处在于，关键词广告出现的位置不是固定在某些位置上，不同的搜索引擎有不同的关键词广告，有的将付费关键词的检索结果排列在搜索结果最前面，也有出现在搜索引擎页面的专用位置。竞价排名的好处是马上可以获得巨大的访问量，但是要为每次点击付费，排名越高，价格越贵，所以只能是一种作为短期目标来进行的搜索引擎营销方式。

2. SEO

SEO是近年来较为流行的网络营销方式，也是搜索引擎营销的一种方式，通过优化Web应用结构、Web页面代码和内容等方式，帮助搜索引擎的搜索程序找到含有最佳内容的Web页面，提高Web应用在搜索结果中的自然排名。简单地说，SEO是一种让Web应用在百度、Google、雅虎、AOL和AltaVista等搜索引擎获得较好的排名从而赢得更多潜在客户的网络营销方式。

Web页面在搜索结果中的排名，直接关系着其访问量。在Google搜索结果中排名第1的页面能获得排名第2的页面两倍的访问量，而且只有约5%的人们点击搜索结果的第2页。SEO的主要工作是在了解各类搜索引擎如何抓取Web页面、如何进行索引以及如何确定其对某一特定关键词的搜索结果排名等技术的基础上，对Web页面进行相关的优化，以提高在搜索结果中的排名，从而提高Web应用访问量，最终提升Web应用的销售能力或宣传能力。

SEO营销策略是一种把SEO效果发挥到最大的一种手段。常见的SEO策略有内容优化、关键词、链接、Web应用布局、Web应用架构、域名和主机优化策略等，其中内容优化和链接策略是SEO的核心策略。

1）内容优化策略

Web应用的实际内容是Web优化策略的一个重要的因素。如果要让Web应用能在搜索结果中排名靠前，Web应用中的内容必须经过精心编写和优化。Web应用内容质量的好坏，直接关系到搜索引擎的搜索程序记录信息量的多少，因此，内容充实是Web应用成功进行SEO策略的基本需要。用户在查找信息时，总是希望找到一个包括很多重要信息的Web应用，很自然，一个页面内容丰富的Web应用要比那些页面内容不那么丰富的Web应用排名要好得多。无论是搜索引擎还是访问者都希望看到比较新的信息，所以不要忘记及时更新内容，做好内容的优化工作。

2）关键词策略

关键词指的是潜在客户或目标用户在搜索引擎中为寻找其所需内容而输入的语句，它

是搜索引擎优化的基础。SEO的目的之一就是提高页面与某个关键字之间的相关性。用户在搜索引擎中搜索资料时，大多数是采用关键词的方式，因此一个Web应用是否更方便被用户搜索到，Web页面关键词的合理设定将起到非常重要的作用。然后针对关键词密度、相关度、突出性等进行优化。

关键词策略需要在文章中频繁提及关键词，一般采用如下步骤。首先，确定Web应用的核心关键词，一般为5个左右，哪一个词或者哪两个词能够最准确地描述Web页面的内容，哪一个字或词被用户搜索次数最多；其次，核心关键字定义上的扩展，例如核心关键词的别名、仅次于核心关键词的组合等额、核心关键词的辅助等；第三，研究竞争者的关键词，分析一下排名占有优势的竞争对手的Web页面，他们都使用了什么关键词，等等。最后，做好关键词的布局，关键词可以从上到下、从左到右，应该无处不在。

关键词出现在Web页面标题标签(<TITLE>)中；URL里面有关键词，即目录名、文件名中可以加一些关键词；在Web页面导出链接的链接文字中包含关键词；用粗体显示关键词(如关键词)；图像ALT标签可以放入关键词；整个文章中都要包含关键词，但最好在第一段第一句话就放入；在元标签(<META>)放入关键词。建议关键词密度最好在5%～20%之间。

3）链接策略

链接策略是通过有计划地优化Web应用中的内部链接和外部链接，以增加反向链接。如PageRank(PR)是Google用于评价一个网页的重要性的一种方法，而影响PR值的关键因素包括导入链接质量和导出链接数量等方面。

内部链接是指Web应用内部页面间的链接关系，除了直接决定Web应用的逻辑结构和影响搜索引擎对Web应用页面的收录外，还影响Web应用中每个页面的权重和相关性。搜索引擎可以根据Web应用内部页面间的链接关系统计出页面的得票数，从而计算出每个页面的内部链接权重。内部链接策略应该注意如下几点。

(1) 控制文章内链的数量，穿插于文章内的链接可以根据内容的多少控制在3～8个左右。

(2) 链接对象的相关性要高。

(3) 给重要的页面更多的关注，使重要的更有价值的Web页面得到更好的排名。

(4) 使用绝对路径或者采用URL重写实现动态连接的静态化。

(5) 采用重定向技术进行URL标准化。

外部链接是指本Web应用以外的链接，表达不同Web应用之间的链接关系。与内部链接相反，外部链接具有不可操控性，即Web应用所有者不能通过正规的手段操控本站以外的页面的链接数、链接对象及链接目标。因此，Web应用所有者并不能操控页面的外部链接权重，这也使得外部链接在决定页面权重及相关性方面起着至关重要的作用。外部链接策略，通常可以通过交换链接、制造链接诱饵、投放带链接的引文等方法来建设外链。

4）页面布局与框架策略

根据项目的特点，优化Web应用的栏目、内容和关键词等方面布局和组织。HTML标签的使用、关键词密度和位置、URL命名等页面布局在SEO中占有很大的权重。要注重页面逆向优化，而大型Web应用的页面优化价值大多数呈现逆向顺序，如：最终页＞专题页＞栏目页＞频道页＞首页；长尾关键词＞热门关键词＞核心关键词＞主打品牌。而对于带框

架的页面需要一定的优化技巧，由于在框架页面中，搜索程序看到的只有一个页面，那就是框架页，这就限制了搜索引擎索引到更多的内容。因此，使用框架页面的时候，可通过改写代码，如：<noframes><body>点击查看Web应用地图</body></noframes>来完成框架页面的转化和优化。

5）域名策略

域名是Internet网络上的服务器或网络系统的名字，在Internet范围内是独一无二的，是Internet中用于解决地址对应问题的一种方法。作为网络时代的“环球商标”，域名是Web应用在网络中能被其他用户找到的直接入口，而搜索引擎对于域名及主机的识别规范，也会直接影响Web应用在搜索引擎中的排名。

为了让Web应用所在的域名容易被搜索引擎检索到，首先域名的命名应该简洁并且有一定的内涵和意义，例如企业的名称、产品名称、商标名或品牌名等；其次，应该拥有自己的独立域名而不是二级域名；最后就是对域名后缀的选择，通常认为“.org”、“.net”、“.cn”、“.com.cn”等后缀有着更高的排名优势。

合理利用网络实名、通用网址以及其他类似的关键词网站快捷访问方式来实现Web应用推广的方法。使用自然语言和Web应用的URL建立起对应关系，例如选择企业名称或者商标、主要产品名称等作为中文网址，这样可以大大弥补英文网址不便于宣传的缺陷，便于用户记忆，更容易体现品牌形象，也更便于用户从搜索引擎结果中发现Web应用。

6）主机优化策略

托管页面的主机性能对SEO也存在影响，要选择性能稳定的托管主机。Web应用的访问者希望被访问的目标站点能够被快速地打开，同样，当网络机器人或网络爬虫正要光顾一个目标Web应用时，如果目标站点主机正处于瘫痪状态，那么对于Web应用来说可能会有两种后果：一是失去将被网络爬虫获取的机会；二是被网络爬虫判断为不存在，将从搜索引擎的索引数据库中删除，所以一定要使用稳定的托管主机。

另外需注意的是，在更换服务器前，要确保目标主机上的页面目录和文件与更换前的主机保持一致。在测试通过后，再进入域名管理系统，修改DNS，指向新主机的IP地址。域名解析一般需要几小时甚至更长的时间，在此期间用户可以正常访问站点的页面内容，即使网络爬虫此时访问，也会抓取更新的页面内容。同时可以考虑采用重定向技术，将旧用户引导到新的URL地址。

SEO是一项系统的工作，包含两个方面的内容：一是Web应用内部因素，如内容、功能、服务等；二是Web应用外部的因素，如友情链接、关键词的外部链接等。在这两个方面里，内部的因素比外部的因素更重要。要做好SEO，得先做好Web应用自身的建设。Web应用的结构合理、内容充实、功能完善、思路清晰是做好SEO的前提，特别是能给用户提供服务才是Web应用的生存之道，SEO只是一个辅助手段而已。

9.2.3 病毒式营销

所谓病毒营销（Viral Marketing）是指通过消费者向消费者传递营销信息的行销方式，是一种基于社会网络的营销手段，通过有意或无意的设计，促使信息在互联网用户之间通过口碑传播渠道相互传播，从而达到信息迅速扩散的效果，类似于计算机病毒的传播。也就是说，通过提供有价值的产品或服务，“让大家告诉大家”，通过别人为你宣传，实现“营销杠杆”

的作用。病毒式营销已经成为网络营销最为独特的手段，被越来越多的商家和Web应用成功利用。

社会网络为病毒营销提供了更为广阔的天地，很多人通过BBS、博客等方式建立了更广泛的社会关系。与传统的信息传播方式相比，Internet提供了低廉的传播成本，这种低成本不受地域限制，为病毒营销大范围、跨地域传播提供了可能。在病毒营销活动中，消费者是信息传播的主体，承担了几乎全部的营销工作。

病毒式营销是基于营销概念的重大变革而产生的，与传统干扰式营销不同，它多以诱导为方式，在宣传产品的同时更主要的是给予客户同类产品的选择、使用、养护、鉴别真伪等相关知识，同时还为消费者提供可参与的娱乐活动、沙龙天地等聚居场所，让消费者之间通过不断的相互影响来自发传播营销信息。也就是说，以消费者主动对产品进行宣传为基础，做好吸引人的内容，让消费者去扩散传播。病毒式营销的营销思想核心就是给人们乐意倾听的理由，然后创造一种进行概念传播的机制。

病毒式营销的成功案例有Hotmail、Amazon、ICQ等国际著名网络公司，其中以Hotmail的经典营销方式最为突出。作为世界上最大的免费电子邮件服务提供商，在创建之后的一年半时间里，Hotmail就吸引了1200万注册用户，而且还在以每天增加15万新用户的速度发展。然而在用户数量激增的同时，Hotmail并没有花费很高的营销和广告费，原因在于Hotmail利用了“病毒式营销”。在每一封邮件的结尾处，Hotmail都附上一句“现在就获取您的Hotmail免费信箱”，吸引受众点击链接注册新邮箱。因此，每一个用户都成了Hotmail的推广者，这种邮件中的推荐取得了惊人的雪球效应。

进行病毒式营销一般可以通过以下几种方式运行。

1）免费的服务

一些大型的Web应用或公司会提供免费的二级域名、免费空间、免费程序接口、免费计数器等资源，这些资源中可以直接或间接地加入公司的链接、其他产品的介绍或广告。由于这些服务都是免费的，对用户有着很大的吸引力。另外，当用户自己在使用并对外宣传时，就也为提供该服务的公司做了免费宣传。

2）便民服务

便民服务比较适合小公司或个人Web应用。在Web应用上提供日常生活中常会用到的信息查询，如公交查询、电话查询、手机归属地查询、天气查询等，把这些实用的查询集中到一起，能给用户提供极大的便利，会得到用户很好的口碑，也就能很快地在网络中推广开来。

3）节日祝福

每当到节日时，可以通过QQ、MSN和E-mail等工具向朋友发送一些祝福，后面跟上Web页面地址或精美图片。因为在节日里，大家都很高兴收到来自朋友的祝福和喜欢发祝福给朋友，一个病毒链就这样形成了。

4）精美Web页面或笑话

娱乐是人生活的追求，工作的目的也是为了更好的生活。做一个精美的Web页面或精彩的笑话发给朋友，朋友一定会很高兴，极有可能会很快地将这一个Web页面或笑话发送给他的好朋友。

5）口头传递

网络上使用最普遍的口头传递方式是“告诉一个朋友”或“推荐给你的朋友”等。很多

Web应用在网络广告、新闻信息和电子邮件后面使用类似的语句。对这种方法，各种Web应用的使用率不同。对于一些娱乐性的Web应用，“告诉一个朋友”的使用率可能会高些。但对于大型Web应用，这类语言的使用率主要取决于所推荐内容的类型和用户群的特点。这种病毒式营销启动成本低并能快速执行，其效果还可以通过引入竞赛和幸运抽签得以增强。

6）利用人际关系网络传播信息

人际关系网络是利用六度理论的一种有效方式，由家庭成员、朋友或同事构成。六度理论的通俗解释就是在人际脉络中，要结识任何一位陌生朋友，这中间最多只要通过六个朋友就可以达到目的。

每个人都生活在人际关系网络中，根据社会地位的不同，一个人的人际关系网络中可能有几十、几百甚至数千人。互联网的用户同样也在发展虚拟社会中的人际关系网络，他们收集电子邮件地址，建立邮件列表与众人沟通，通过洽谈室结交新的朋友。认识到实体社会和虚拟社会中这些人际关系网的重要作用，通过病毒式营销把信息置于人们的各种关系网络之中，从而迅速扩散出去。

7）通过策划事件营造传播话题

策划事件是事件营销（Event Marketing）的核心，策划运作一个大范围或局部（或行业范围、圈子范围）轰动的事件，引起消费者、媒体和社团的兴趣和关注，以提高Web应用的知名度，其特征在于迎合时代人的心理需求，如好奇心、欲望、需要、贪念、贫乏等。通过网络，一个事件或一个话题很容易地进行传播和引起关注。

9.2.4 Web 2.0推广

在Web 2.0时代，突出的不是纯技术，而是参与和互动。Web 2.0定位非常清晰，针对特定的用户群，有着自己核心的业务。因此Web 2.0的来临，使得多种针对特定用户的全新互动式网络广告载体（如BLOG、RSS等），受到了极大的关注。

Web 2.0营销是指对博客、RSS、WiKi、SNS等Web 2.0应用技术和理论的一个综合营销表现。其核心是注重用户的交互作用，让用户既是网站的浏览者，也是网站内容的建设者。由于用户能够方便地畅达地为自己所消费的产品表达意见，因此这些内容先天具备再次推广产品的价值。

应用Web 2.0技术进行推广也是比较有用的推广方式，常用方法有通过博客、RSS和百科类站点。

(1) 博客营销：Web应用企业通过博客与消费者进行交流沟通，发布企业新闻，收集反馈和意见，实现企业公关，等等，达到增进客户关系，改善商业活动的效果。博客营销有低成本、分众、贴近大众、新鲜等特点，往往会形成众人的谈论，达到很好的二次传播效果。

(2) RSS营销：利用RSS技术向用户传递有价值的信息来实现网络营销目的的活动。Web应用企业利用RSS技术可以及时地把最有价值的商业信息“推”向用户，使用户不必每天去访问成百甚至上千的网站，就可以获取这些网站最新的信息，从而使企业更为有效地开展网络营销活动。

(3) WiKi营销：通过WiKi类站点，在上面回答问题，然后留下Web应用的链接，等等。

Web 2.0 催生网络营销的革新，并将以此不断激发用户产生强烈的互联和消费意愿。个性化、针对性和创新性的营销手段是 Web 2.0 时代网络营销发展的必然，是传统营销方式的提升、飞跃和创新。

Web 应用推广是互联网营销取得成效的基础，同时 Web 应用推广的效果表现在多个方面。Web 应用推广营销要综合考虑多种相关因素，应当注重 Web 应用推广的系统性和阶段性。在制定 Web 应用推广策略时必须明确 Web 应用推广的目的，如为企业创造价值。在同一 Web 应用发展的不同阶段，Web 应用推广采用不同的策略组合的系统性思想，根据企业内部资源条件和外部经营环境来制定，并且对 Web 应用推广各个环节、各个阶段的发展状况进行有效地控制和管理；应当基于其 Web 应用推广工作的目标、预算等各种推广方式进行取舍，灵活地构建一套适合自身需要的成本低、效果佳的有针对性的 Web 应用推广解决方案，积极和持续地开展多层次、多样化甚至整合众多方式的 Web 应用推广，努力把 Web 应用和产品及服务推介给尽可能多的现实和潜在的顾客，从而为自己创造更好的效益。推广营销只是一种手段，Web 应用要想长期取得好的成绩，提高自身的服务质量才是最重要的。

9.3　内容维护

对于一个 Web 应用来说，决定其是否能吸引用户的关键因素是其是否具有高质量的内容，如文本、图片、音频和视频等。由于 Web 应用自身特性，只有信息及时、内容充实的 Web 应用才是用户喜爱的 Web 应用，才能在第一时间吸引用户并使其驻留。版面的调整、内容的更新和完善，以及活动的推广等都是内容维护必不可少的方面。因此，在 Web 应用启动运行之后，需要对内容进行良好的维护，需要采取一些措施以保证有专门团队或第三方内容维护服务提供商进行内容编辑，如工作流定义和分发策略；技术方面应该支持非技术人员对内容进行编辑；应该支持全球化访问，即多种语言的内容。

Web 应用内容维护主要包括如下几个方面。

1）适时更新内容

了解用户需求，保持用户的忠诚度，适时更新 Web 应用内容是一个重要的方法。不同类型的 Web 应用，确保内容即时性的周期也不同。对新闻系统来说，更新周期要非常短，一旦有信息更新就应该立即更新，可能每个小时都有更新。对于企业来说，当企业推出新产品、新的服务内容，逢年过节、喜庆的日子，或更新了企业内部活动图片、企业动态新闻，等等，都应该第一时间在 Web 应用上反映出来，以便让客户、合作伙伴、用户、员工等利益相关者及时地了解相关详细状况，这种更新周期可能是季度、月、周甚或天。而对于电子商务类应用而言，还需要考虑单个内容的流通需求并满足这些需求。旅游类应用中可用信息和价格信息是其在线预定交易的重要内容，其更新要求很高，不正确的价格可能会导致客户取消预定，过期的可用信息导致客户预订酒店或者超过酒店容纳量，等等。

通过使用内容管理组件，使得当内容管理发生时，同步更新或修改 Web 应用展示的内容，也就要求动态实时生成或短周期地生成 Web 页面。

2）栏目撤换

一个 Web 应用中的栏目，可能在 Web 应用构建初期有非常美好的设想，但是随着时间

的推移，发现它并不是一个受用户欢迎或者无法适应最新内容的栏目，这时，过时的栏目就可以撤换，或者因特殊事件而增加栏目，如西安电子科技大学要迎接2011年校庆而增加校庆栏目。撤掉是容易的，但建设新栏目需要格外慎重。开设新栏目需要认真规划，考虑有没有稳定的高质量的内容来源，是不是用户喜闻乐见的形式，以及增加栏目会增加多少工作量，等等。

3）Web页面更换

Web页面是用户和Web应用互动的主要平台，更换Web页面可以让用户耳目一新，如西安电子科技大学为了迎接校庆而更换页面展示风格。Web页面的更换首要目标是使得Web应用的访问更加有效和友好。例如，有的Web应用页面把所有的栏目的更新都列在各自栏目内容的顶部，每个栏目列出最新的5条更新。结果发现该Web应用的更新频率跟不上，每个星期只能有一个栏目更新。时间一长，虽然每个星期都有栏目在更新，但是Web应用首页看上去像没有更新一样。因为更新的内容淹没在旧内容里了。这时，可以把所有的更新都集中在一个地方，这样可以使得更新显得比较快。

4）社区维护

一般Web应用都提供用户与用户、用户与Web应用组织者之间进行交互的虚拟社区，如各种BBS(Bulletin Board System，论坛)、聊天室等。而大多数BBS逐渐向多媒体、多功能、虚拟现实和角色扮演的方向发展，以发帖和回帖为主要形式的BBS仍然是所有社区功能中最重要、最核心的部分，它已经成为一个传播新闻、消息、观点的重要渠道。对BBS的维护，主要是整合和梳理BBS上杂乱无章的信息，依靠完善的管理技术和相关规定管理BBS。做好BBS的维护，主要需要关注以下几个方面的内容。

(1) 定位好BBS，合理规划好版块。

(2) 制定清楚规则。BBS中的规则是BBS质量的基础。对诸如水贴、广告贴和恶意用户的管理都是需要制定规则并且严格执行的。BBS规则的执行力，也是BBS质量的保证。

(3) 明确社区成员层次，保持成员平衡增长。BBS是一个社团，社团有社团的游戏规则：如果一个社团进新人的速度超过了自身的消化能力就会带来颠覆规则的危险。

对于这种存在的论坛管理，一般采用如下几种技术手段来进行管理。

(1) 定时关闭开放技术。

(2) 先审后发技术。

(3) 关键字过滤技术。

(4) 积分(经验值)限制功能。

(5) 封堵用户名或IP技术。

9.4 Web使用挖掘

在Web应用系统运行的过程中，会产生大量的日志和各种数据记录。分析和挖掘Web访问日志和用户数据可以帮助Web应用的运行维护人员理解用户的行为和Web应用的使用方式，从而大大提高Web应用的运行维护工作的效率。

Web使用挖掘也称Web应用的使用分析，指评估和分析Web应用使用的各种指标来度

量Web应用是否成功的过程。通过从服务器日志、代理服务器日志、浏览器日志、用户数据、会话数据、Cookies等网络监控文件中抽取用户访问Web应用的浏览模式、页面的访问频率以及隐藏的用户感兴趣的信息等，挖掘出潜在的用户行为特征和规律，预测用户的浏览行为，为用户提供个性化服务、信息导航、Web应用结构的改进与优化，以及为商业智能提供依据。

根据应用的不同，可以将Web使用挖掘分为两种主要领向：一般访问模式跟踪和定制使用跟踪。一般访问模式跟踪通过分析Web访问日志来理解访问模式及倾向，利用这些分析可以清楚地给出较好的Web结构及资源提供者的分组情况。把数据挖掘技术应用于Web访问日志可以获取有效的访问模式，这些访问模式有助于网站的重构，指出Web页上有效的广告位置及研究特殊的用户行为和特殊的销售广告，等等。定制使用跟踪可以分析个人的倾向，它的主要目的是为每个用户定制符合其个人特色的Web站点。根据个人喜好，可以在显示的信息、网站结构及资源格式等方面动态地进行定制。

Web使用挖掘框架通常如图9.2所示，即主要分为数据采集、数据预处理、模式发现和模式分析4个阶段，如图9.3所示。

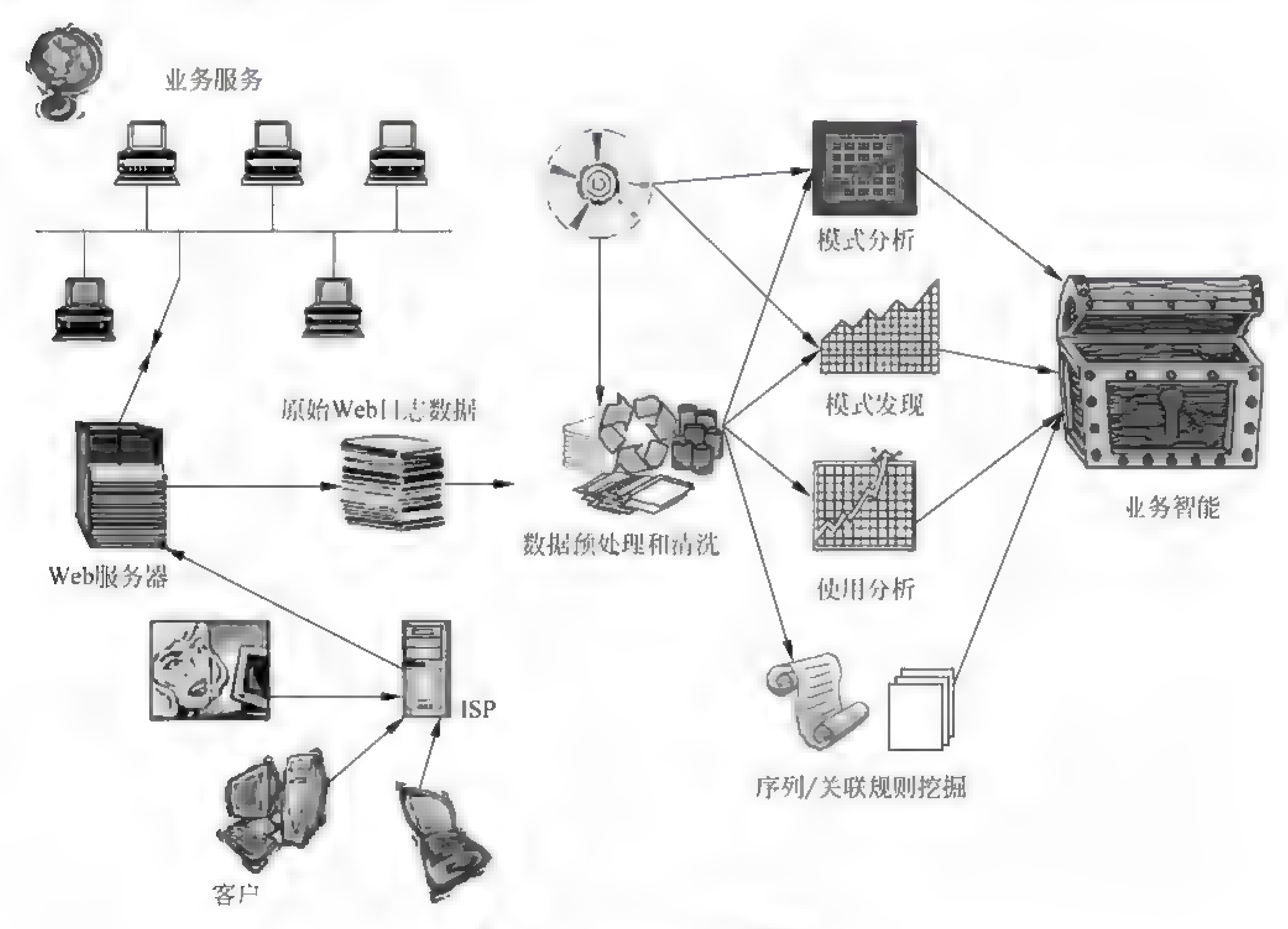


图9.2 Web使用挖掘框架

1. 数据采集

Web使用挖掘是从Internet上的海量数据中挖掘出潜在有用的信息，首先需要获得可供挖掘的数据，构造实现目标任务的数据集。由于HTTP是一个无状态的协议，很难准确得到用户的浏览信息，所以需要从Web的结构出发，多层次地从Web服务器端、应用服务器端、客户端、代理服务器端等进行Web应用相关信息采集，形成后续挖掘的数据集。

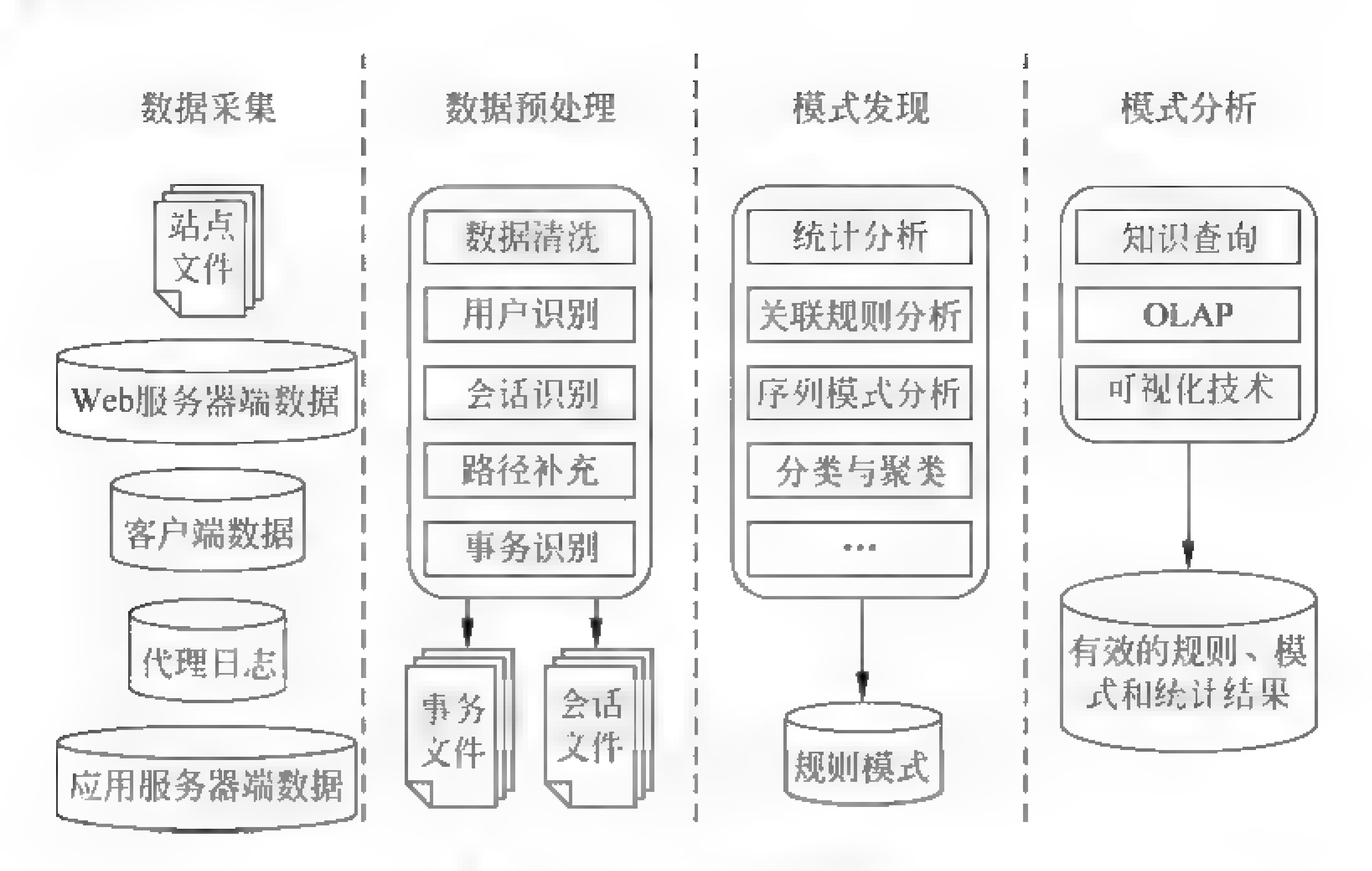


图 9.3 Web 使用挖掘流程

1）Web 服务器端数据

服务器日志记录了服务器接收处理请求以及运行时错误等各种原始信息，记录了用户访问 Web 应用的数据，如下所示：

```
192.168.1.58 -- [17/Jan/2011:18:23:45 -0800] "GET /webe/ch9.htm HTTP/1.0" 304
```

这些数据包括：访问客户的 IP 地址、访问时间、访问的页面、访问方式、页面大小、浏览器类型、响应状态等。通过对日志进行统计、分析、综合，就能有效地掌握服务器的运行状况，发现和排除错误原因，了解客户访问分布，等等，更好地加强系统的维护和管理。常见的 Web 服务器日志工具有 WebTrends、AWStats、Webalizer、Analog、Summary、WebLog Expert Pro 等。

2）应用服务器端数据

这种数据采集方法可以利用应用服务器上的应用程序（如 CGI 程序）来记录用户的个人信息，同时也可以通过自定义的格式动态记录用户的浏览信息。每次用户进行访问时先进行身份验证，然后由应用程序记录用户全部的浏览过程。这种采集方法与 Web 服务器端的数据采集方法相比，在用户确定方面的准确性高，但是大量的应用程序会使系统的效率很低。

3）客户端数据

客户端的数据收集优于直接从 Web 服务器日志文件收集用户使用数据，原因是这种采集以用户浏览行为数据为基础，可以准确地捕获用户的浏览行为、浏览路径和浏览时间，避免了用户识别、会话识别、路径补充等繁琐的数据预处理过程。

客户端的数据收集可以使用诸如 JavaScript 或者 Java Applet 等充当远程代理来实现，也可以修改用户的浏览器软件，使之具有数据收集的能力。利用远程代理的方法能自动将访问信息上传到 Web 服务器上。通过修改浏览器的源代码加强浏览器的功能也可以使之具有收集访问信息并上传的能力。

客户端还可以以 Plug-in（插件）方式嵌入监控程序。Plug-in 以后台操作方式随时跟踪页面的浏览情况，浏览器可以在一个页面上或同时在几个打开的窗口上用_New 加载同一

个插件的多个实例。当用户离开实例所在的页面或关闭窗口时，用 Destroy 删除插件的实例，并把相应的浏览信息以 Socket 方式送到服务器端，这种方法的缺点是浏览器要安装 Plug in。

从客户端收集数据最大的优点是可以直接取得用户的各种真实信息，这些信息的完整性和真实性都要优于服务器上的信息。但是，因为直接从客户端取得数据需要考虑用户的隐私和占用用户的机器及网络资源，所以客户端的数据收集特别需要用户的合作。

4）代理服务器数据

通常在 Web 中基于安全和效率的考虑，需要使用代理服务器技术。代理服务器在用户端和服务器端扮演着中间传递的角色，而且代理服务器可以多级级联。代理服务器通常为多个用户服务，这样从代理服务器上就可以得到多个匿名用户的浏览信息。代理服务器上保存着一个最近访问过的页面集合，如果这些页面是静态的，那么用户通过代理服务器访问该页面时，就不需要从 Web 服务器上取得数据，可以将该静态页面直接发给用户。但是对于电子商务中经常使用的动态页面就要从 Web 服务器上取得所需数据。

Web 使用挖掘数据采集技术汇总如表 9.1 所示。

表 9.1 Web 使用挖掘数据采集技术汇总

采集特点		采集类型			
		Web 服务器端	应用服务器端	客户端	代理服务器端
用户	单			√	
	多	√	√		√
站点	单	√	√		
	多			√	√
使用强制性		否	否	是	是
面向整个 Web		否	否	是	是
影响 Web 服务		是	是	否	否
采集时间准确		否	否	是	否
准确识别用户		否	是	是	否

2. 数据预处理

数据预处理是保证 Web 使用挖掘质量的关键因素。没有经过数据预处理的数据集是海量的，要想在这样杂乱无章的海量数据集中挖掘出用户感兴趣的模式和规则是不可能的。因此，首先要将原始数据集进行数据预处理，形成完整的用户会话；其次再将会话分割为适合挖掘的用户事务；最后采用相关的挖掘算法在用户事务集上挖掘有用的用户信息。显然，数据预处理结果的好坏直接影响到挖掘用户潜在且有用的模式。数据预处理包括以下几个方面。

1）数据清洗

数据清洗又称数据精简，就是删除采集的源数据中与 Web 数据挖掘不相关的数据，并转化为适合挖掘的可靠的精确数据，比如剔除一些不必要的图片信息等，此外，剔除一些明显出错的访问记录。通常预先定义规则库来帮助删除记录。

典型的 Web 服务器日志记录了包括用户 IP 地址、用户 ID、请求的 URL、请求方法、访

问时间及日期、传输协议、传输字节数、错误码、参照页和用户代理等属性，其中与数据挖掘相关的只有用户的IP地址、用户ID、请求的URL、访问时间及日期等信息，其他辅助信息均可忽略。此外，还需要从日志中清理出包括出错记录、图像文件请求记录以及网络爬虫的浏览记录，因为网络爬虫对Web应用的访问不带任何感情色彩，不能反映用户的访问特性，而出错记录则属于噪音，需要被清除。

2）用户识别

用户识别是将用户与请求的页面相关联的过程，目的是从服务器日志中区分出不同的用户。常用的方法有：不同的IP代表不同的用户，通过浏览器和操作系统识别用户，Cookie方法。由于本地缓存、代理服务器和防火墙的存在，使可用数据不完整，有效识别用户的任务变得十分复杂。一般采用基于日志站点的方法，还可以使用一些启发性规则。

3）会话识别

会话指一个用户在进入Web应用到离开Web应用这段时间的一系列页面请求，会话识别则是将用户在一段时间内访问某Web应用的页面分解为单个会话。在跨越时间区段比较大的服务器日志中，用户可能会多次访问该Web应用，会话识别的目的就是将用户大量的访问记录划分为单个的会话。会话识别中典型的方法是时间阈值法。时间阈值法是通过计算在两个请求的时间间隔是否超过一定的阈值来识别会话的边界。除此以外，还有会话切分与合并算法、基于用户兴趣度的时间阈值法等。

4）路径补充

由于客户端页面缓存技术和代理服务器的广泛使用，使得Web服务器访问日志所记录的可能不是用户完整的访问路径。当用户向Web服务器发出页面请求时，被请求页面已经存在于缓存或者代理服务器中，那么这个请求就不会传送到Web服务器，在Web服务器访问日志中就不存在本次请求的记录，从而使得Web服务器访问日志不完整。不完整的访问日志不能准确地反映用户的访问模式，对这样的日志进行挖掘得到的模式可能不全面、不准确，因此，有必要进行访问路径的补充。

路径补充是将由于本地缓存或代理服务器缓存所造成的遗漏的请求页面补充完整，利用Web应用的拓扑结构，对页面进行分析，如果根据Web应用的拓扑结构发现用户当前请求的页面与用户上一次请求的页面之间没有超链接关系，就出现了路径不完整情况。检查访问日志，如果这个页面的参照页面在用户历史访问记录之中，可以认为用户使用了“后退”按钮。根据Web应用的拓扑结构发现，在用户访问的历史记录中，有多个页面可以直接链接到该页面，那么将选定在请求时间上最接近当前请求页面的页面作为本次请求的参照补充页面，进行路径补充。补充了页面后，还需要估计补充页面的访问时间，一个简单的方法是取两个页面访问时间的中间点作为补充时间的访问时间。

5）事务识别

事务识别是为了把用户会话分成对用户有意义的组，建立在对用户会话识别的基础上，目的是依据数据挖掘任务的需求将事务做分割或合并处理，以利于知识的发现。事务识别的常用方法有三种，时间窗口法、最大向前参考法和参引长度法等。

事务识别方法中最简单的方法是时间窗口法，即定义一个时间长度，该时间片内用户浏览的所有页面均归为一个事务。

比较常用的事务识别方法则是最大向前参考法，它是根据用户访问行为来划分会话，一

旦用户后退浏览已经浏览过的页面时，就找到了事务的边界。具体做法是，从用户访问的首页开始，到第一个回退动作为止定义为一个事务，接下来的第一个向前动作引发下一事务，直到下个回退动作产生。周而复始，将用户访问页面序列划分为一个个事务。

除此之外，参引长度法(Reference Length)也是一种常见的事务识别方法。该方法认为，一个用户在某一个页面所停留或者使用的时间取决于该页面对于他来说是内容页面还是辅助页面，因此该方法也被称为内容辅助页面方法。如果已知内容和辅助页面的集合，在顺序读取日志记录时，一旦遇到内容页面就找到了事务的边界。

3. 模式发现

模式发现是从预处理之后的数据集中挖掘出潜在模式与规则的过程。经过数据预处理之后，原来杂乱无章的用户访问日志转变为能够进行挖掘的用户会话或者用户事务，此时，就可以用模式发现的相关方法和技术从中挖掘出潜在的规律和模式。模式发现的主要方法有统计分析、关联规则、序列模式、聚类、分类以及依赖建模等。

1) 统计分析

统计分析是从 Web 应用中获取知识的最常用的方法，通过 Web 统计工具统计出最常访问的页面、页面的平均访问时间、平均访问路径的长度、有限的错误分析等有关 Web 应用使用的基本信息，分析用户对网络内容的关心程度。常用的统计分析方法有判别分析、因子分析、相关分析、回归分析、偏最小二乘回归方法等。目前有很多 Web 流量分析工具，如 Logan Pro、WebTrends、WebLog Expert 等。

2) 路径分析

路径分析可以被用于判定一个 Web 应用中最被频繁访问的路径，利用这些信息可以改进页面之间的链接关系以及 Web 应用的结构。通过路径分析可以得出类似如下的信息。

①访问 Web 应用的用户中有 25%是从页面 B 开始的。

②访问 Web 应用的用户中有 10%的用户访问的前 4 个页面是：A、D、F、H。

③有 15%的用户访问路径是 C→M→D→E。

针对①，可以在页面 B 上直接添加希望传达给用户的各种信息，或者通过链接指向相应的页面，从而提高信息的点击率；针对②，可以在页面 A、D、F 和 H 上添加相应的广告宣传信息，从而提高广告点击率；针对③，可知路径 C→M→D→E 为频繁访问路径，可以在这几个页面上添加其他超链接或者促销信息，从而增加其他信息的访问频度。

3) 关联规则

关联规则是描述在一个事务中事件之间同时出现的规律的知识模式，在优化系统中可通过关联规则分析事务集合中数据项之间的相关联系。关联规则挖掘最著名的方法是 Apriori 算法和 FP-增长(Frequent Pattern Growth)算法。现在还有大量的改进算法，包括并行 Apriori 算法、基于矩阵的 Apriori 算法。

Web 应用的关联规则侧重发现页面内容和链接访问之间的关系，利用所发现的 Web 应用的关联规则，调整页面链接顺序，改进 Web 应用设计，减少等待时间，并可以为特定用户动态调整 Web 应用的结构，预设他可能关心的页面内容，使用户访问的有关联的文件间的链接能够比较直接，让用户能够容易地访问到想要的页面。Web 应用如果具有这样的便利性，能给客户留下较好的印象，增加再次访问的几率。

4）序列模式

序列模式是在会话集中发现有时间序列关系的模式，即各页面存在时间上的先后顺序。如访问了打印机页面的用户，在访问完该页面之后，有60%的人又访问了扫描仪页面。在Web服务器日志中记录了用户访问页面的到达时间，这样就可以得知用户访问页面时间的先后，在此基础上可以通过序列模式挖掘算法发现具有时间顺序页面间的关联关系，为用户推荐下一个访问的页面提供了一定的依据。

以下4个规则中，①和②是序列模式，③和④是关联规则。比较①和③，②和④便不难发现，①和②考虑了访问的先后顺序，③和④则没有考虑时间因素。

① 访问页面P1之后有30%的用户又访问了页面P3。

② 访问页面P1和P3之后有35%的用户又访问了页面P5。

③ 访问页面P1的用户中有30%也访问了页面P3。

④ 访问页面P1和P3的用户中有35%的用户也访问了页面P5。

序列模式挖掘著名的算法为AprioriSome、AprioriAll算法，也有很多改进算法，如序列链接的改进算法、基于前缀树的增量序列挖掘算法等。

5）聚类

聚类用于将事务集合中相似的数据项归并为若干个类。聚类本身并不知道有多少个类别，是根据相似度将Web页面聚为不同的类别。在Web应用优化系统中，聚类可分两种方式：基于共同浏览行为的用户聚类与基于相似内容的页面聚类。

Web用户聚类就是将具有类似访问浏览模式的用户归类到一起，如聚类中的用户在访问一个新闻站点时经常读取娱乐和时尚方面的报道，这类知识有助于为用户提供个性化信息服务，或在电子商务中进行顾客信息统计以划分顾客集，其主要算法有Leader、EM、BIRCH、Autoclass和COBWEB等。Web用户聚类还包括，通过基于用户会话的Web页面聚类算法，发现用户最频繁访问的页面组和页面路径，将该路径加载到未包含它的页面中以此优化Web应用结构，并为运行维护人员优化Web应用站点结构提供有力的依据。而Web页面聚类则可以按内容之间的相关性建立具有相关内容的页面组，可用于页面搜索等相关操作上，主要算法有PageGather、K-means（K平均值）和K-centers（K中心点）等。

6）分类

分类是将Web页面划分到预先定义好的不同类别中去。分类中的类别、属性和特征都是预先定义的，将剩下的页面划分进不同的类别。比较常用的分类方法有决策树分类法、支持向量机分类法、K-临近分类法和贝叶斯分类法等。其中决策树分类法与其他分类方法相比，具有很多优点，如速度快，容易转换成简单的、便于理解的分类规则，容易转换成数据库查询语言，以图形化的方法展示分类效果，有较好的直观性。

在Web使用挖掘中，通过模式发现的分类算法，可以发现用户感兴趣的页面类别，因而可以向该用户推荐该类别中尚未被访问的页面，方便用户查找所需的信息。

7）依赖建模

依赖建模是Web挖掘中另一类重要的模式发现方法。其目的是建立模型来表示Web领域的各个不同因素之间的重要依赖关系，例如建立一个用户网上购物过程的模型，那么这个模型要能够表现出用户在电子商店挑选商品所经历的各个阶段。常用方法有隐马尔可夫模型（Hidden Markov Model，HMM）和贝叶斯网络（Bayesian Belief Network，BBN）。Web

使用模式依赖建模不仅为分析 Web 用户行为提供了一个理论框架，而且对预测 Web 资源消耗也有很大的帮助，同时可以用来对电子商务站点提高产品销售或对用户访问提供决策支持。

4. 模式分析

模式分析阶段是从模式发现阶段所获得的众多模式和规则中提取用户感兴趣的模式和规则。在模式发现阶段挖掘出的大量模式和规则中，有相当多的部分不是有效的，或者是用户不感兴趣的。因而，Web 使用挖掘需要采用相应的技术和方法进行模式的提取、分析，将结果最终汇集成用户真正感兴趣的、潜在有用的知识。常用的模式分析方法有知识查询、OLAP 和可视化技术等。

1）知识查询

由于模式发现阶段得到的大量规则会使用户不知所措，借助知识查询机制，根据用户的查询目的，自动搜索相关的模式和规则，帮助用户从大量的模式和规则中查询出用户自己感兴趣的、对自己有用的和真正需要的模式和规则，帮助分析用户的目标。达到这一目的的途径有两个：①在挖掘之前，在数据库上设置约束，使挖掘只在一部分数据中进行；②挖掘过程中执行查询语句，不断筛选出需要的信息，将无用的数据过滤掉。在 SQL 的基础上，研究人员提出了适用于 Web 数据挖掘的查询语言，如 WebSQL、WebLQM、Squeal 和 DMQL 等。

2）OLAP

OLAP(On-Line Analytical Processing，联机分析处理)是基于数据仓库系统的多维分析工具，在多维数据模型(如星型模型、雪花模型、事实星座模型等)的基础上，用不同的格式提供数据，能够满足用户的不同需求。OLAP 技术先要构造多维数据立方体，然后在该立方体上通过上卷(Rool-up)、下钻(Drill-down)、切片和切块(Slice)等操作从不同角度分析数据，在不同层面上发现知识。

OLAP 具有灵活的分析功能、直观的数据操作和分析结果可视化展示等突出优点，从而使用户对基于大量复杂数据的分析变得轻松而高效，以利于迅速做出正确判断。OLAP 专门设计了用于支持复杂的分析操作的功能，侧重对决策人员和高层管理人员的决策支持，可以根据分析人员的要求快速、灵活地进行大数据量的复杂查询处理，并且以一种直观而易懂的形式将查询结果提供给决策人员，以便他们准确掌握企业(公司)的经营状况，了解对象的需求，制定正确的方案。

3）可视化技术

Web 数据挖掘的可视化技术是指使用 GUI 来帮助用户挖掘和理解大量的复杂数据，采用图形化方式将更加直观，更容易理解，建立了用户与数据挖掘系统良好的交互通道。例如，可以通过图形化方式或根据对不同重要程度的知识采用不同的颜色来显示，将用户感兴趣的知识明显地显示出来。

可视化技术主要分为挖掘过程的可视化、挖掘结果的可视化和知识管理的可视化三类。挖掘结果可视化主要是应用在模式分析阶段，通过图形的方式展示挖掘的结果，比如以点的方式展示数据对象的统计值，也可以用二维、三维的图形来显示序列的转换，等等。

IDL(Interactive Data Language，交互式数据语言)是面向矩阵、语法简单的第四代可视

化语言，它支持 OpenGL 图形加速、量化可视化表现、集成数学与统计学算法、方便的数据输入输出方式、跨平台图形用户界面工具包、连接 ODBC 兼容数据库及多种程序连接工具等，是目前科学数据可视化方面较好的工具。

9.5 总结与展望

Web 应用并非一次开发就能一直运行，Web 应用自身的特性，使其开发并非一蹴而就，运行维护阶段的代价非常大。通过 Web 应用的运行，可以发现 Web 应用测试中仍然未能发现的问题，方便 Web 应用工程师完善 Web 应用，进而提高 Web 应用的质量属性。Web 应用往往需要大力推广才能获得更多的关注，因此，Web 应用的维护工作是其生命周期中的一个重要环节，直接决定了 Web 应用的成败与否，只有维护良好的 Web 应用，才会吸引用户的访问。本章从 Web 应用内容维护、推广营销和使用挖掘等多个方面对 Web 应用的运行与维护进行了详细的分析与阐述。

内容的及时更新和用户反馈的有效管理是 Web 应用内容维护的主要内容。内容更新主要包括：周期地更新静态网页，管理动态生成的网页的内容，页面与栏目的调整和 BBS 维护等内容。Web 应用的推广营销除了采用传统的广告模式之外，还有网络广告、搜索引擎营销、病毒式营销以及 Web 2.0 等方式。选择何种营销模式需要根据 Web 应用自身的特点来决定。Web 使用挖掘还面临不确定性（如动态 URL）、可伸缩性（Web 日志巨大）、动态性（用户使用数据始终在变）、从点击到概念（内容概念化、分类、网站本体）以及实现推荐系统等方面的诸多挑战，也是目前研究的热点之一。Web 使用挖掘目前的研究主要集中在如下 4 个方面：①数据预处理集中日志本体的构建，新的 Log 格式和挖掘语言的构建，服务器、代理和客户端日志集成，用户和会话识别；②模式发现算法的改进和海量数据挖掘；③模式分析，包括可视化、OLAP 和知识查询；④在应用领域关注移动互联网、网络安全、社交网络和个性化的推荐系统。

随着 Web 技术的不断发展，对 Web 应用的运行与维护也提出了更高的要求。研究知识管理和语义 Web 技术，并与 Web 内容管理相结合，提高机器可读和可理解性，实现高效准确的 Web 内容管理，是提升 Web 应用运营和维护的重要方式之一。在语义 Web 的基础上，结合自动化挖掘和度量技术，实现自动化的 Web 应用使用挖掘是目前研究的重点和难点之一。

第10章 Web项目管理

随着Web应用开发技术的不断发展和用户对Web应用功能、性能、安全性、可用性等要求的不断提高，Web应用的设计已经不能再仅仅简单地利用静态HTML文件来实现，不再仅仅只由一两名Web页面设计人员单独创建而成，Web应用的设计与开发进入了需要强调流程、分工、团队合作的时代，建立规范的、有效的、健壮的开发机制，才能适应用户不断变化的需要，达到预期的计划目标。

Web应用的开发周期短等特点，使项目面临各种挑战，同时，Web项目与传统软件开发的不同，使Web项目管理与传统软件项目管理存在不少差别，导致传统的软件项目管理方法不完全适用于Web项目管理。

项目管理是有关人管理其他人的行为的活动，这种以人为中心的活动需要项目经理具有很强的解决冲突能力，需要Web开发团队之间具有很好的跨学科理解能力。Web应用开发的模型必须具有灵活性，以适应高度迭代、增量开发，并和客户方便沟通。这就使Web项目管理更多地从传统项目管理转向敏捷方法，并需要在整个项目周期中更好地使用集成工具，进行风险管理和配置管理。

10.1 Web项目管理面临的挑战

软件项目管理就是为了使软件项目能够按照预定的成本、进度、质量顺利完成，而对人员(People)、产品(Product)、过程(Process)和项目(Project)进行分析和管理的活动，目的是为了让软件项目尤其是大型项目的整个软件生命周期(从分析、设计、编码到测试、维护全过程)都能在管理者的控制之下，以预定成本按期、保质地完成软件交付用户使用。其主要任务包括领导、开发和监控。软件项目管理先于任何技术活动之前开始，并且贯穿于软件的整个生命周期。

Web应用由于其特性，使得Web项目管理与传统的软件项目管理存在着差异，如表10.1所示。如在传统的软件系统中通常是按功能分组，而这些功能分组是关键的度量指标。而在Web应用中，功能和内容互相依赖，同等重要，其共同可用性构成必不可少的因素。

随着Internet和Web应用技术的高速发展，大量的应用软件都基于Web开发，同时大量的传统应用开始向Web环境移植，但是开发人员的开发方式却还停留在Web发展的早

表 10.1 软件项目于Web应用在项目管理方面的区别

主要参数	软件项目	Web应用
主要目标	花最少的钱开发最优质的产品	花最短的时间开发一个可用的产品
项目大小	一般比较大,需要花费10～100甚至更多的人参与	团队经常比较小,一般3～9个人
持续时间	一般12～18个月	一般3～6个月
花费	较大,一般百万数量级	较小,一般以千计量
开发方法	基于需求,结构化,迭代进行,文档驱动	敏捷方法,集成组件,原型化开发
技术	面向对象方法,CASE工具	基于组件的方法,可视化编程,多媒体
过程	CMM,ISO等	敏捷过程
产品	可重用性比较差,而且都是比较复杂的应用	可重用性高,标准组件,很多标准的应用
人员组织	很多都是有着丰富经验的软件开发专业人员等	多媒体设计人员,Web应用开发人员,市场推广员等

期阶段。开发人员通常用一种粗糙的、随意的方式进行Web应用的开发,经常会出现比如个人英雄主义,不重视设计文档等的不规范开发流程,缺乏严密的系统的技术、有效的方法、严格的管理和相应的质量保证机制。Web应用的开发、配置和管理方式也不注重开发前的准备工作,项目一开始就直接进行开发与部署,导致很多情况下Web应用不能按期完成,或者Web应用构建完成后达不到用户的期望要求,使得客户不满意,造成Web应用修改不断,最终尽管耗费了很多人力和资金、费用超出预算,仍然很难构建出用户满意的Web应用。

Web项目管理主要任务是管理和协调项目,合理分配和使用资源,保证Web项目按计划顺利进行。需要进行以下几方面的工作:组织和管理开发团队;制定Web项目的版本和迭代计划;评估项目中的各种风险,并对风险进行监控和管理;对Web应用的资源和版本进行配置管理,等等。还需要综合考虑交互设计、展示设计、内容设计和功能设计的相互映射与有机结合,以及所有这些设计方面对产品角度的协调。Web项目在考虑传统软件项目管理普遍会遇到的管理问题外,还需考虑以下特有问题。

1. 领导挑战

Web项目管理面临着领导挑战。Web应用因其独有的Web特性,大多还处于从头开始开发,缺乏经验,难以合理评估费用,因此,Web项目管理面临着巨大的重用挑战。

Web项目管理很多情况下受技术特长者的控制,致使忽略了传统软件开发的组织,而且在计划的时候,很容易出现过度乐观的情况,这种乐观经常得到市场和销售人员的支持。

和传统软件相比,在Web应用开发项目中,不清晰、不完整的计划目标,计划目标频繁变化,以及项目组织存在缺陷等问题更多。

2. 开发挑战

Web应用开发是一项充满智慧的工作,所以在Web项目管理中,存在着很大的挑战。

1）开发人员个性鲜明

很多软件开发项目更像是一门艺术，程序员一般都很有个性，他们的能力也相差很大，大多个性突出、自我意识浓厚，更倾向于自由、多样的工作方式，这就导致很难评估实际的人力需要。更严重的是，很难将这些个人主义者融入一个组织中。由于Web应用是智力和劳动密集型项目，受人力资源影响较大，项目成员的结构、责任心、能力和稳定性对Web应用的开发造成了极大的挑战。

2）新颖性

Web应用用户未知，使得开发人员需要不断应对不断变化的需求。Web项目经理需要靠直觉谨慎权衡用户兴趣和不安全需求。另外，由于大多Web应用是新开发的，所以，难得有经验可借鉴。

3）大量候选方案

传统软件开发中，一个特定的问题有大量的解决方案，很多情况下，不可能预先比较和评估这些不同的方案。而Web应用开发中，因为很多组件和半成品可以被使用，所以，候选方案的问题可能会稍微小一点，但是也不应该低估。

4）快速而持续的变更

对Web应用来说技术变更是非常典型的问题。计算机软硬件技术的快速发展使得计划和组织项目变得更加困难。尽管一个大型的项目开发经常按计划进行，可是新的、更好性能的组件却进入了市场，这意味着项目在进行中，需要改变设计。

Web应用构建过程中，客户的需求不断地被激发，并不断地进一步明确，这也直接导致Web应用开发计划、预算金额等处于持续的变更之中。

5）动态性

Web应用面临很大的上市压力、短开发周期的压力、高更新频率的压力、激烈的竞争压力，致使敏捷方法、组件系统、集成开发工具等被大量使用。可以通过将Web应用拆分或分解成多个子项目进行开发，以更好控制和管理小的子项目开发组。

6）并行开发

Web应用的拆分或分解使多个子团队并行开发，而这些子团队和传统软件项目团队组织结构有非常大的差异。在传统软件开发中，通常是面向开发进行拆分，如分为GUI开发、数据库连接、算法建模等小组，即这些小组可以根据技术特长进行分配。而在Web项目中，因为这些组件具有类似的功能和设计结构，只是针对不同的客户特性，所以分组之间除了交流组件接口之外，还需要交流组件的功能和设计结构。

Web项目管理要保证具有类似经验或能力的开发人员分配在不同的小组以及不同小组之间的顺利沟通，以确保Web应用开发作为整体进行，避免重复开发。而这类沟通在传统软件开发工具中难以映射。

7）持续性

Web应用本身及其所用工具持续演化，导致难以实现从开发到维护的转移和内容增长等。需要通过定义增量过程模型（每个增量可能只有几周甚至几天）来管理这种持续演化。另外，Web应用需要24×7提供服务，维护面临更大的困难，因此，必须利用配置管理工具，记录需求和设计决策以确保开发路径和变更历史可重现和可理解。

8）不成熟性

Web应用开发人员相对年轻，经验不足，但同时又对新技术和新工具有极大的热情和

兴趣，喜欢尝试新的工具和技术，更新最新版本。如尽管目前HTML5还没有出现最终版，很多人就已经在尝试使用，并用Firefox4.0 Beta版来支持HTML5。这种尝新的态度会导致除了开发人员的爱好之外，无明确理由地在一个组织中使用大量工具和技术。Web项目管理需要利用这些热情并选择合适的技术和工具，需要规定变更政策和更新策略，并监控其执行程度。

除了开发人员的热情之外，Web应用环境也不断更新，导致出现错误、界面扩展等方面的问题不断增长。许多不成熟的工具由于没有其他工具可选而必须使用。因此，采用快速更新的技术开发在面对人员经验不足的同时，还阻碍着Web应用的演化。针对这些问题，Web项目管理可以借助大的工具和技术提供商，这些大的提供商会有一些规律的更新和版本发布策略以及完善的服务。如GNU环境或者Apache组织等可靠的开源项目也是不错的解决办法。

3. 监控挑战

项目经理往往很难控制"无形的软件产品"，难以确定软件产品到底实际完成了多少，同时程序员也很有可能遮掩发展的真实状态。而由于Web应用具有功能和内容并行开发的特征，所以，对于客户和项目经理来说产品更"有形"。此外，由于Web应用开发迭代周期短，更易于检查。因此，监控挑战在Web应用中相对次要一些。

4. 使用挑战

对于Web应用及其使用，Web项目管理面临着如下特殊的挑战。

1）看似简单

几年前，Web应用开发给人以非常简单的印象。实际上，早期的静态Web页面如果不考虑链接逻辑开发，可以那么认为。而对现在复杂多样且具有大量软件功能的Web应用而言，它包含复杂的处理逻辑，数据可能在数据库或数据仓库中，信息及时动态生成，同一浏览器中调用多个应用。因此，很难评估开发和处理代价。

2）艺术性

Web被认为是最能感知时尚的软件，Web页面也更容易更新以符合时尚趋势，致使对Web应用运行和维护阶段有更多更频繁的变更。其实更多时尚的变化大多集中在静态的数据，如Logo的变化，因此，Web项目经理对这些特定领域计划应该具有足够灵活性和动态，使得艺术的变化仅仅局限在相应的内容，而不用改动程序代码。

3）用户自发性

Web应用的用户对Web应用无忠诚可言，对Web应用的使用也不愿阅读使用指南，因此，Web应用应该具有自我解释性，使用户在无指导手册的情况下就能轻松使用Web应用所提供的逻辑，即Web应用具有可用性。在计划阶段怎么强调Web应用的可用性都不过分。可以通过采用一些Web应用可用性评估工具辅助Web项目管理。

4）普适性

Web应用的全球可访问特性，使得无法预知实际用户的特性及其使用位置，也就无法在Web应用开发时识别这一需求。通过部署试运行有时并不能获得不管是批评还是赞同的反馈意见。

5）兼容性

浏览器产品众多，如微软的IE、Google Chrome、Mozilla Firefox、360浏览器等及其不同版本，对标准的支持程度各不相同，加之浏览器的使用配置的决定权在用户，相互之间兼容性也就不明确。Web项目管理可以借助一些测试平台测试Web应用在不同浏览器中的支持程度。

6）稳定和安全

21×7的可用性意味着用户对Web应用以及底层软硬件的质量要求很高，而且要求Web应用对用户的访问隐私进行保护。这就对Web应用的维护提出更高的挑战。Web项目管理就需要借助一些诸如WinRunner等的配置管理系统来提供基本保障。

7）可伸缩性

Web应用的用户自发性和普适性，意味着在开发阶段难以规划其扩充性。为了保证不挫伤用户访问积极性、不造成数据丢失，Web项目管理需要考虑软硬件的在线扩展。

10.2 Web项目人员管理

Web应用日益复杂、难以依靠个人力量构建完成，必须依靠团队的力量，按照技术领域分工合作、各司其职，以完成复杂Web应用的开发。Web应用开发是一项以人为中心的活动，强调个人能力与团队协作有机结合的重要性，团队成员之间的有效沟通、Web项目经理的有效协调和高效管理，都是Web应用项目成功与否的重要因素。

Web应用的团队成员一般都具有以下几个方面的特性。

（1）多学科性。因为Web应用一般由内容、超文本结构和用户界面等组成，且往往面向不同领域，所以Web开发人员一般都需要具有不同特定领域的知识。

（2）并行开发。Web应用一般由问题域划分成子项，由于其专业性，Web项目组的组成结构都是大体相似的，这意味着有许多并行开发需要协作进行。因此，在Web应用中需要比传统应用软件花费更多的时间进行交流沟通。

（3）团队规模较小。因为Web应用开发周期短，而且项目预算有限，所以Web项目团队大多在10个人左右。大型的项目会被划分为子项目，然后由子团队并行开发。

Web应用的以上这些特性，使得Web应用项目管理者在人员管理方面必须更关注项目的团队组织和团队管理。

10.2.1 团队组织

构建一支优秀的项目团队，是Web应用成功的前提，而加强项目团队协调和领导，组建一支背景广泛的团队是建立高效项目团队的前提。团队组织首先要对整个团队组成人员框架进行设计，除考虑每个人的教育背景、工作经验外，还需考虑其兴趣爱好、个性特征以及年龄、性别的搭配，确保团队队员优势互补、人尽其才。其次，Web团队也必须满足敏捷开发思想中拥抱变化的特点，组建敏捷团队。

Web应用项目团队中的人数视情况而定，应该适应于Web应用的需求和预算。人数过多或过少，会使Web应用不能及时交付或增加不必要的沟通代价，会带来管理的不便。

在Web应用项目团队中，考虑不同阶段对人员的不同需求，设定多种角色及其任务。每个成员至少扮演一种角色，清楚自己在项目组中所扮演的角色及其任务，承担起责任。一般包含项目经理、策划人员、系统管理员、页面设计人员、开发(传统功能开发和多媒体开发)人员、测试人员、维护人员以及领域和业务专家等角色。

1) 项目经理

项目经理是一个项目的核心，负责管理一个成员能力参差不齐的项目组团队，团队成员有着不同的教育背景、习惯和价值观等。项目的成功与否，很大程度上取决于项目经理的协调、管理和控制能力。项目经理主要担负的职责可以分为对内和对外两部分。

对项目团队内部而言，项目经理主要是对项目进行管理和执行，包括：项目总体设计，开发进度的定制和监控，定制相应的开发规范，负责各个环节的评审工作，协调各个成员(小组)之间开发。

对外，项目经理要同时面对客户和其他项目组。项目经理要做好客户的维护和交流工作，及时、全面、准确地了解客户的需求和变化，并对需求变化进行控制，确保项目的进度和计划得到良好的执行。项目经理要同其他项目经理交流、协调，在合理范围内尽可能维护、争取和配置资源，确保本项目的顺利进展。项目经理更多体现为项目的推动者、服务者、领导者和协调者，尽可能地推动项目成功。

Web应用项目经理应该遵循一些规则，Herwig MAYR给出Web项目经理的如下10条黄金规则。

① 鼓励团队成员，使其保持对项目的高度热情和士气。

② 强调不同领域知识的重要性。

③ 快速解决冲突，没有人总是赢家，输家也不总是同样的人。

④ 经常给团队成员解释其所承担的角色和职责。

⑤ 设定明显的并行开发并使用尽可能的协作。

⑥ 将文档工作公平分配给相关任务承担者，并承认文档工作的同等重要性。

⑦ 从项目一开始就鼓励和协调一致采用开发工具。

⑧ 将工作规模的重要指标进行对照转换。

⑨ 催促客户持续参与项目。

⑩ 始终留意项目的进度和项目目标。

2) 策划人员

策划人员的主要职责是详细了解用户需求，广泛调查研究，提供详细、合理的需求分析和策划方案。在某些重要项目中，策划还应当从用户体验的角度出发，为用户使用方式和方法提出策划方案和指导意见。此外，策划还应包括后期Web应用推广方面的策划。

策划人员以客户需求为依据，以调查研究为指导，策划导航、结构、信息和内容等方面。客户在提出需求时，往往会发生许多问题，例如需求不全面、需求范围和目标过大、需求不合理甚至提不出需求等，这些都需要策划人员同客户进行深入的沟通，在调查研究的基础上，对客户需求进行引导和梳理，使需求分析能够尽可能地完善，也使项目组有能力完成，同时为需求变更管理打下基础。

3) 系统管理员

系统管理员主要负责根据Web应用开发人员和系统运行环境的要求对服务器进行管

理、配置和维护，涉及软硬件操作的技术技能、网络与通信技能。此外，系统管理员还应当对Web服务器安全、运行安全、运行性能、数据安全和数据备份负责维护。

4）页面设计人员

页面设计人员主要完成Web应用的展示设计。客户对于项目的认可程度和满意程度，往往就是关注界面是否美观，是否符合期望，使用是否方便，等等。

页面设计人员除了要有扎实的美学功底和创意外，还要具备Web应用易用性设计的概念和意识。在项目执行过程中，页面设计过程往往也被划分为前中后三个阶段。前期主要根据策划和用户对象对界面进行分析，完成界面的模型设计，并能形成总体概要型的Demo提供给客户，供其选择和接受；中期主要完成界面的详细设计，根据策划方案设计详细平面稿，同时考虑美观、功能和易用；后期则主要对界面进行修饰，并完成相关动画的设计制作。

5）开发人员

开发人员的主要工作是开发Web应用各项功能。在具体的设计开发环节中，根据Web工程的要求，开发人员又可以分为系统架构设计师、模块设计师、程序员、数据库管理员、测试人员以及多媒体人员等多种角色，并依据科学的流程进行开发。在常见的Web应用中，开发人员的主要责任包括如下内容。

(1) 分析用户需求并且为Web结构设计程序。

(2) 确定训练和教育的区域。

(3) 建立维护与反馈程序。

(4) 为Web应用的运作设计并实施相关策略。

(5) 为Web应用实现安全性和访问权限。

(6) 为数据库交互、服务器端和客户端设计程序并建立文档。

(7) 按照要求发布Web页面。

(8) 将Web应用部署在相应软硬件平台。

6）测试人员

测试人员负责在项目进行过程中对Web应用进行测试，尽可能地发现并排除Web应用中存在的缺陷和漏洞。

7）维护人员

在Web应用部署后，根据客户要求负责Web应用的更新和维护工作。一般情况下Web应用的维护需要多种人员参与其中，包括负责服务器维护的人员、负责内容更新的人员、负责推广营销的人员和负责用户使用性分析的人员等。

8）领域和业务专家

Web应用开发过程中，鼓励客户在整个开发过程中都在场，目的是为了及时交流需求，明白客户目的。也意味着很多和领域及业务有关的内容，只有领域和业务专家才更清楚。领域专家和业务专家是构建领域知识和业务过程的最佳人选，他们对Web展示和对业务目标的全局以及客户服务等方面具有重要作用。

10.2.2 团队管理

优秀的Web应用开发团队建立在合理的开发流程及团队成员密切协作的基础上，Web应用项目团队成员共同迎接挑战，有效计划、协调和管理各自的工作以完成明确的目标。有

效的团队协作需要依靠有效的团队领导和协调，每一个成员具有自我管理和自我领导的能力，彼此以沟通、持续反馈为原则，通过合作、互补方式，达到迅速应变的效果，最终完成团队的共同目标。

团队成员工作是否积极，需要团队负责人用一些有效的激励办法进行调动。团队负责人应该通过协调管理、精神和物质奖励等手段，最大限度地提高团队成员的工作热情，进而提高Web应用的开发与维护效率。

Web应用团队负责人责任重大，除了需要具备调兵遣将的能力外，还应该更加努力地参与Web应用所有方面的工作。这样才能够了解每一个环节出现的问题，包括内容写稿、Web应用维护、活动策划等。发现了问题，也能即时进行修正。而且在每个环节的交流中，应该和相关的团队成员进行沟通。

只有提高了团队成员的生产效率，才能提高Web应用的开发与维护效率。在团队管理方面，可以参考以下几个方面的原则。

1）合理配备，各尽其才

根据成员的数量、质量和专业等特点安排相应的工作，使每个成员都能发挥自己的特长，而又不至于影响整个项目的进度或质量。这样有利于激发成员的工作积极性和创造性，也有利于在项目组成员之间形成和谐融洽的气氛，从而提高工作效率，保持项目组的稳定。

2）培养团队精神

即使项目成员来自不同的组织和公司，也要强调每个项目成员对于正在参与的项目的共同理念，要强调个人职责所在的项目背景，强调项目共同目标，就项目进展情况、项目风险以及其他的管理问题以相同的方式进行交流，共享经验和教训，增强团队的凝聚力。使团队成员相互支持、互相交流和互相尊重，也是培养团队精神的重要内容。

3）建立良好的工作环境

Web应用项目的成功完成需要良好的工作和交流环境，很多具体问题都需要项目经理组织相关开发人员，在融洽的交流环境，通过交流、反馈的手段协商解决。

4）制定良好的规章制度

Web应用项目规模越来越大，项目经理需要通过制定规矩、制定标准来实现制度管人。一个强劲的管理者首先是一个规章制度的制定者。规章制度也包含很多层面：纪律条例、组织条例、财务条例、保密条例和奖惩制度等。好的规章制度可以使执行者能感觉到规章制度的存在，但并不觉得规章制度是一种约束。

优秀的Web应用开发团队还应该具有并遵循共同的工作规范标准。例如，项目管理应该产生规范的项目开发计划，设计评审应遵循统一的文档及审评标准，源代码应符合程序规范条例，测试应产生有规范且可推理的测试计划及测试报告，等等。

5）建立明确的共同目标

Web应用项目团队中扮演不同角色的项目组成员由于地位、视角等的不同，对Web应用的目标、期望值也会有很大的区别。好的项目经理善于捕捉成员间不同的心态，理解他们的需求，帮助他们树立共同的奋斗目标。项目经理的协调使得团队的努力形成合力。

可以通过采用激励的方式来帮助项目组成员建立共同的目标，主要的激励方式有薪酬激励、目标激励、情感激励和授权激励等。

6）及时沟通

每个人的知识结构和能力有着巨大的差异，导致对于同一问题的认识很可能出现相应的偏差，这就需要团队成员之间、团队与团队之间、团队与客户之间在Web应用开发过程中进行及时的沟通，及时地将开发过程中遇到的问题进行讨论与分析，尽可能早地将开发过程中的问题发现并解决，从而降低Web应用开发的风险。

及时沟通包括三个方面的内容。

（1）与用户保持沟通，邀请用户直接参与关键的评审过程，充分了解用户需求，不断修正、完善开发过程。

（2）项目管理者、实施者和验证者之间的沟通，创造相互理解、相互配合的良好工作氛围，激发开发人员的工作热情。

（3）不同开发小组之间、不同软件工程师之间的沟通，及时交流信息，避免工作冲突和不一致。

Web应用开发的需求多变性，使得项目经理与公司高层、与客户之间的沟通能力极其重要，良好的沟通能力将有助于解决这类复杂问题。

10.3 Web应用项目计划

Web应用项目计划是一项系统活动，通过把Web应用分解为多个增量，分析增量的复杂性，将工作量分配给Web应用开发团队成员，规定完成各项任务的起止日期，并严格控制Web应用的开支成本，从而将Web应用的开发时间、成本预算等控制在一个合适的范围内，最大限度地保证Web应用能够在规定的时间和预算内顺利交付。

10.3.1 进度管理

进度管理的目的是保证开发过程按照规定计划进行，最终在规定的日期内能够开发出用户满意的Web应用产品。由于Web应用开发周期短，使得进度管理对开发进度的影响更为明显。通过使用快速准确的计划优化方法，在目标、工期和费用约束前提下得到最有效的项目进度计划，平衡协调各相关部门人力、物力和资金，提高各执行部门的工作效率，从而达到缩短开发周期，提高管理水平的目标，保证Web应用按时保质完成。

Web应用的进度计划将随着项目的进展而不断演化，需要通过比较实际状态和计划之间的差异，并依据差异做出必要的调整，以使项目向有利于目标达成的方向发展。对于Web应用的进度计划可以在项目的早期制定一个宏观的进度安排表，随着项目的发展，把宏观进度表中的每个条目都精化成详细进度表，完成一个活动所必须实现的特定任务要被标示出来，并安排好实现这些任务的进度。

1. 进度管理的主要内容

在Web应用项目开发前，开发项目计划，在一定程度上给出每项活动何时开始和何时完成的日期，并说明每项活动要求多少资源。这种详细的计划称为项目进度表。创建项目进度表是进度管理的核心内容。进度管理主要体现在下面4个方面。

1）制定活动计划

制定活动计划是创建项目进度表的第一步，确定需要执行什么样的活动以及用什么样的次序执行这些活动。项目经理在对 Web 应用进行管理时，首先需定义 Web 应用过程，然后基于 Web 应用过程制定 Web 应用项目计划，根据过程定义中的活动来分配任务，如采用 UWE 进行建模活动。也就是说，这一部分最主要的工作目标是列出 Web 应用（或其增量）开发中所有的活动和任务。

2）工作量估计

工作量的估计采取基于已完成项目的历史数据和专家经验判断相结合的方法。为了对软件系统进行准确估计，常用方法是使用特性分解结构，将 Web 应用分解为可以根据历史数据和个人经验进行较准确估计的所有迭代。如根据项目组是否有采用 UWE 进行建模的经验或已有项目中采用 UWE 进行建模的活动所需的时间进行估计。

对于 Web 应用开发而言，设计方法并不成熟。而随着 Web 应用越来越复杂，设计就成为 Web 应用开发中重要的步骤，如采用 UWE、OOHDM、WebML 等进行设计的人员有限，展示方式也不相同，所以设计工作量是 Web 应用项目工作量的很重要的一部分。因此，在项目工作量估计时，需要认真分析和估计建模和（或）设计工作量。

Web 应用开发进度表中任务尽量是细粒度的，工作任务和较小的里程碑应该按“天”来安排进度，这种细粒度有助于更准确地估算和预测可能的进度延误。

3）编制进度计划

在活动计划和工作量估计的基础上，考虑某一执行时间的具体活动，分析该活动所需的资源，制定资源分配计划。在为每项活动分配资源的基础上，就可以编制和发布项目进度表，以指示每项活动计划的开始和完成日期以及需要的资源。

Web 应用采用更多的是对版本和迭代安排进度计划。在第一次迭代中，安排宏观进度计划，编制宏观进度表，标识所有 Web 应用增量和项目将要被部署的日期。随着增量的开发，识别出特定的 Web 工程任务，并做出进度安排，宏观进度表上增量的条目被精化成详细的进度表。

Web 应用开发项目的进度计划可以从两个角度制定：①Web 应用的最终发布日期已经确定，而且不可变，这就需要将工作量分配在预定的时间框架内；②Web 应用的最终发布日期只是大致界限，真正的发布日期由开发团队自行组织确定，这种情况下，工作量是以能够最好利用资源的方式来进行分配，并编制计划，制定最终发布日期。Web 应用开发项目面对第一种情况的频率要比第二种情况高得多。

4）跟踪与监控

借鉴个体软件过程（PSP）的计划管理方法，以及极限项目管理，要求开发人员进行小组简单日工作总结和周工作小结，整体开发团队根据项目的周期和进度，定期召开项目例会，以便对项目的进展情况进行监控和跟踪。

跟踪 Web 应用增量开发进度的一种方法是由团队成员提供哪些活动已经完成，但是这种情况受个人理解因素影响较大；另一种方法是确定已经实现了多少用户场景，还剩下多少用户场景。

2. 进度管理的方法

虽然对于 Web 应用而言，项目进度计划和传统项目计划在计划的粒度、针对的内容等

方面都有很大差别，进度管理更多地强调执行，但是传统的项目计划的技术仍然有很重要的作用。常用的传统项目计划技术有甘特图(Gantt Chart)和网络计划技术。

1）甘特图

甘特(Gantt)图是一种简单的水平条形图，它以日历为基准描述项目任务，是在1917年由亨利·甘特开发的。水平轴表示日历时间(如时、天、周、月、年等)，纵轴表示活动(项目)，每条表示一个任务，任务名称垂直地列在左边的列中。图中的水平条的起点和终点对应水平轴上的时间，分别表示该任务的开始时间和结束时间，水平条的长度表示完成该任务所持续的时间，它直观地表明任务计划及实际进展情况。甘特图通过条状图来显示项目进度，以及其他系统进展的内在关系随着时间进展的情况。

甘特图能清晰地描述每个任务从何时开始，到何时结束，任务的进展情况以及各个任务之间的并行性，易于理解。有很多专业工具软件支持绘制甘特图，比如Ganttproject、Gantt Designer和Microsoft Project等。

2）网络计划技术

网络计划技术是以网络图为基础的计划模型，其基本思想是用图来表示组成待执行项目的各种活动之间的顺序关系。它能够反映项目进展中各工序间的逻辑关系，描述各工作环节和工作单位之间的接口界面以及项目的进展情况。网络计划技术的两种基本形式是关键路径法(Critical Path Method，CPM)和计划评审技术(Program Evaluation and Review Technique，PERT)，其核心内容包括绘制网络图、识别关键路径和基于关键路径的优化。

CPM最早是通过分析项目过程中哪个活动序列进度安排的总时差最少来预测项目工期的网络分析，是一种基于数学计算的时间规划管理方法。在项目管理中，关键路径是指网络终端元素(事件)的序列，具有最长的总工期，并决定了整个项目的最短完成时间。关键路径的工期决定了整个项目的工期。任何关键路径上的终端元素的延迟将直接影响项目的预期完成时间(如在关键路径上没有松弛时间)。

PERT假设项目持续时间以及整个项目完成时间是随机的，且服从某种概率分布，可以估计整个项目在某个时间内完成的概率。PERT图与CPM在网络图的画法上基本相同，是一个有向图，图中的箭头表示任务，它可以标上完成该任务所需的时间，图中的结点表示流入结点的任务的结束，并开始流出结点的任务。PERT图不仅给出了每个任务的开始时间、结束时间和完成该任务所需的时间，还给出了任务之间的关系。

表10.2所示为Gannt、PERT和CPM在模型、应用层次、考虑因素和适用项目类型方面的对比。

表10.2 常见的项目计划的技术对比表

	模型	应用层次	考虑因素	适用项目
Gannt	无	决策层 管理层	工程进度	小项目
PERT (面向事件)	概率网络	管理层 执行层	时间控制 工程进度 活动次序	大、中项目
CPM (面向活动)	概率网络	管理层 执行层	工程进度 活动次序 成本、时间	大、中项目

3. 工期优化

工期优化也称为时间优化，其目的是当网络计划计算工期不能满足要求工期时，通过不断压缩关键路径上的关键工作的持续时间等措施，达到缩短工期、满足要求工期的目的。缩短工期的方法主要有以下几种。

1）强制缩短法

采取措施使网络计划中的某些关键工作的持续时间尽可能缩短。强制缩短法的重点是选择哪些工作压缩其持续时间来缩短工期。常用的方法如下。

（1）顺序法：按关键活动开始时间确定先开始的工作先压缩。

（2）加权平均法：按关键活动持续时间长短的百分比进行压缩。

（3）选择法：即计划编制者有目的地选择某些关键活动进行持续时间的压缩。

2）调整工作关系

根据项目的可能性，将某些串联的关键工作调整为平行作业或交替作业。

3）关键路径转移

利用非关键工作的时差，用其中的部分资源加强关键工作，以缩短关键工作的持续时间，使工期缩短。采用这一措施，关键路径可能会不断地发生转移。

4. Web 应用进度追踪和控制

在项目进行过程中，必须不断监控项目的进展，以确保每项工作都能按进度计划进行，掌握计划的实施状况和影响进度变化的影响因素，并将实际情况与计划进行对比分析，必要时应采取有效的对策，使项目按预定的进度目标进行，避免工期的拖延。

在 Web 应用中，关键路径是综合了各方面的平衡后确定的，不能轻易改变关键路径来达到工期缩短的目的，而经常采用调整工作关系的方法来进行工期优化。针对 Web 应用的特点，尽可能在构建过程中并行进行大量测试。

Web 应用进度追踪和控制是指采用定期观测计划检查手段，定期对项目计划执行情况进行较为全面、系统的检查，以发现问题并及时采取措施。可以通过不断调查 Web 应用开发团队以确定哪些活动已经完成，这种方法实用但是可能不可靠，其不可靠的原因是很多功能或工作产品只有当其完成时才能明确其进度指标。或者检查哪些用户场景已经实现，还剩下多少用户场景需要实现，这种方法提供了一个完成情况的粗略指标。在工作进度表粒度够细的情况下，可以按照确定完成多少工作任务，以及开发人员对完成工作的信心来确定。通过跟踪建设的实际进度，比较出实际进度是超前还是滞后计划进度，进而分析产生偏差的原因，形成项目报告，反映原因，提出对策。

进度跟踪计算以 WBS 为基础，通过先前经验或专家意见赋予每项工作一个权重，而总权重为 100，跟踪各项工作的实际完成百分比情况，将工作权重乘以百分比得出实际进度，将各项工作的实际进度求和便得出 Web 应用的构建进度实际值。

10.3.2 成本管理

成本管理是指在满足项目质量、工期等合同要求的前提下，对项目实施过程中所发生的

费用，通过计划、组织、控制和协调等活动实现预定的成本目标，并尽可能降低成本费用的一种科学的管理活动。项目管理的主要控制要素是质量、进度和成本，要求在保证质量的情况下，寻找进度和成本的最优解决方案。

Web应用有着各种各样的目的，但是尽量节约成本，提高收益仍然是Web应用一个主要目的。Web应用成本是指在Web应用建设和运行维护过程中所发生的资金耗费。对于Web应用而言，Web应用建设规模的大小、功能要求及作用的差异，也会使不同的Web应用间的成本差异极大。如直接在Internet上架设自己独立的Web应用，通常要申请专用线路，配备功能齐全的设备及专业技术人员，一般要花费几十万甚至数百万的费用。而寻找合适的网络服务提供商，申请网上主机服务，就不需要额外的配备；由于资源共享，每个客户主机用户所承受的硬件费用、Web应用维护费用、通信线路的费用均大幅度降低，每年的花费只在几千元至几万元之间。上述两方面原因使起着同一作用而实现方式不同的Web应用其建设成本不同，并由此决定了计算Web应用的成本不能和其他行业那样简单地计算单位产品成本，而应从Web应用的实际投入入手评估Web应用成本的构成并计量。

Web应用成本根据Web应用从建设到运行维护的各阶段可将其划分为Web应用建设成本与运行维护成本两大类。在各类中又可根据费用的用途进行逐级细分，如建设成本可分为硬件成本、软件开发成本和其他成本等。相对而言，硬件成本和其他成本可根据Web应用规模及形式予以估算。系统开发成本却较难确定，因为，其成本估算涉及人、技术、环境和政策等诸多因素，所以，估算方法较为复杂，一般包括以下几种。

(1) 参照已有的同等规模Web应用的开发成本，估算Web应用的开发成本和工作量。

(2) 将整体的Web应用开发成本分解为若干部分，在估算出每部分的成本和工作量后，再汇总估算整体Web应用开发成本。

(3) 将Web应用构建期分解为若干段，分别估算出Web应用开发在各阶段的开发成本和工作量，再汇总估算整个建设期间的系统开发成本。

Web应用运行维护成本指Web应用建成投入使用后，为保障其正常运行需向网络管理机构支付的运行费用以及Web应用的技术维护和管理等费用，包括材料消耗、推广营销、法律咨询及其他开支，若Web应用建设资金为贷款还包括资金利息等。

10.4 Web项目风险管理

诺贝尔奖获得者诺顿有一句名言："不对风险进行管理是最大的冒险。"项目风险管理是一项系统活动，是项目管理组织积极主动地运用各种风险管理技术，通过对项目可能遇到的风险进行识别(Identification)、分析(Analysis)、优先级评定(Prioritization)、应对(Provisioning)、监控(Monitoring)和缓解(Mitigation)，以最大限度地避免或降低风险发生的可能性，减少风险对项目产生的不良影响和损失，保证建设目标的顺利实现。

10.4.1 Web工程风险特性

Web工程除了具有传统软件开发遇到的风险外，它还存在着如下一些特有的风险。

(1) 目标定义不明确。很多开发人员对他们构建的Web应用的目的并没有一个明确的认识。

(2) 错误的目标用户。很多Web应用都过分地强调了公司的管理，却忽视了应用的目标用户。

(3) 新技术风险。例如Web应用开发需要使用HTML5进行开发，但是Web应用开发项目团队没有HTML5方面的经验。

(4) 面向开发的页面结构。开发人员最简单的创建内容的方式就是遵循组织结构图，一旦组织结构图应用到了一个Web应用中，它就不太可能遵循一个面向客户的操作了，Web应用将会重点放在公司而不是用户方面。

(5) 因为外包导致缺少一致性。同一个公司的不同Web应用(或一个大型的Web应用的部分)的开发如果外包给了不同的其他公司，这将是一个高风险，因为这些公司的特性会影响到Web应用，特别是在内容和导航上。每一个公司出于自身的目的都会把自己和其他竞争对手区分开来，这就导致Web应用的外观和操作会存在很大的不同。

(6) 缺少维护预算。传统的软件开发项目大概年度维护费用在10%～15%，因为Web应用内容更新和变更技术，使得其维护费用会更高一些，大概会耗费至少50%的年度开发费用用于维护工作。

(7) 内容回收利用。很多人尝试着从传统的文档资源中获取Web应用的内容，而传统的文件资源都是线性的，但是Web应用是非线性的，所以内容被重新利用的进度会减缓。

(8) 链接结构不好。线性内容的回收利用导致链接经常被人工地增加，这是有害的。而且很多链接指向的是更高排序的页面，而不是特定的内容，这就导致用户需要从起点寻找他们所需的内容，并且有足够的耐心去做。

(9) 混淆Internet和Intranet。Internet是向外界展示一个公司的企业文化方面的内容，来吸引潜在的用户；Intranet是用来使公司的员工有效地工作。

(10) 混淆市场调研和可用性调查。虽然市场调研是评定用户的期望，可用性调查的目标是发现用户如何处理Web应用以及他们存在的问题，但市场调研永远都发现不了一个Web应用所需的特定的功能需求。混淆了这两个方面的内容，很容易导致错误的推断。

(11) 低估了Web的策略性。许多公司都认为基于Web进行展示是理所当然的，长期以来忽略了这个沟通渠道将导致严重的战略竞争缺陷。但是又有许多公司高估了这一战略重要性，因此在开发Web版本时导致失误。

另一方面，Web应用安全性面临着特有的安全性风险。OWASP (Open Web Application Security Project)在2010年更新排名前10的安全风险，依次为：注入(Injection)、XSS(Cross-site Scripting)攻击、认证和会话管理失效、不安全的对象直接引用、伪造跨站请求(CSRF)、安全配置错误、不安全的加密存储、限制URL访问失败、传输层保护不足，以及未经验证的重定向和转发。除了这排名前10的安全风险之外，也不能忽略可能给Web应用造成的安全性危险的其他风险，可以从开发、测试已经编码等方面进行考虑。

Web工程团队应该从两个不同层面考虑风险：一个是风险对整体Web应用项目的影响，如对交付时间、开发价值的影响，变更对进度的影响，新技术新方法对团队成员的影响，技术成熟性和可能的变化，等等；另一个是风险对当前正在设计的Web应用版本的成功部署的影响，当前版本是否需要新技术，团队技能组合能否胜任当前版本，等等；以及Web应用可能面临的安全风险。

任何Web项目都需要时刻关注最关键的风险，以确保这些风险可以被降低和消除。评估和控制这些风险是Web项目风险管理的主要任务，如图10.1所示。

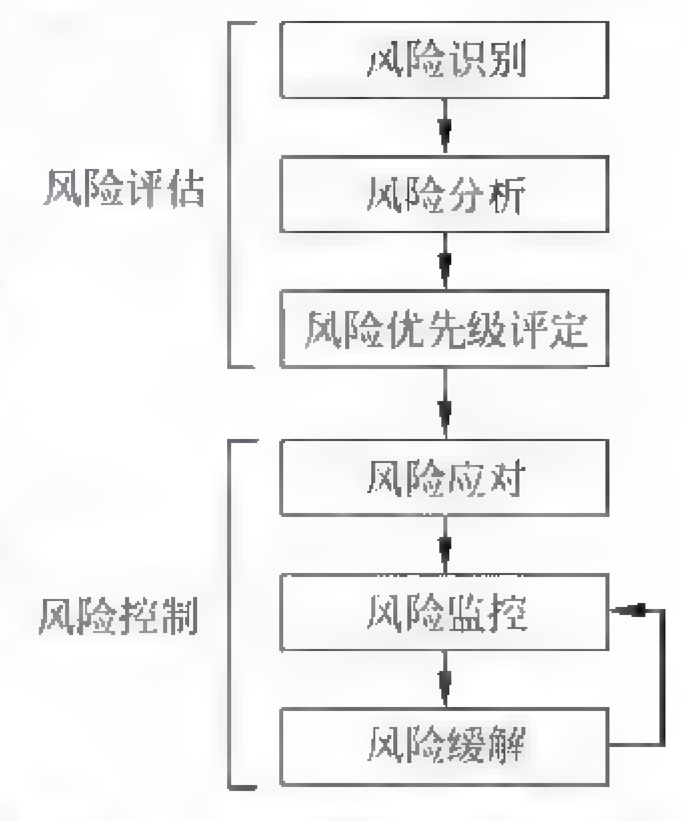


图10.1 风险管理的主要任务

10.4.2 风险评估

如图10.1所示，风险评估的主要任务是风险识别、风险分析和风险优先级评定。

1. 风险识别

风险识别是项目风险管理的基础和重要组成部分，试图系统化地确定那些可能发生并危及项目成功的条件、情况或者事件，以全面、系统地识别潜在风险，合并类似风险的规避措施。风险识别结果应该包括风险的来源、分类、表现以及其后果。

1）风险来源

风险来源标识了风险可能发生的常见领域。常见的内部和外部风险来源有：不确定的需求、无法达到的工作量估算、不可行的设计、不成熟的技术、不切实际的进度估算与安排、人手或技能不够、成本与资金问题、不可靠的或不充分的分包商能力、不可靠的或不充分的供应商能力，以及与实际的或潜在的客户或他们的代理沟通不充分，等等。

2）风险分类

风险分类有利于对风险进行识别和归纳。可以从多个角度进行分类，如按项目生命周期模型的阶段、按过程类型、按项目管理的风险和从风险的不确定性角度等分类。从风险的不确定角度可以将风险分为已知风险、可预测风险、不可预测风险。通过识别已知的风险和可预测的风险，使得Web应用项目管理人员能够估算风险产生的影响，进而便于避免和控制这些风险。通常将风险分为人员风险、产品风险和过程风险。

风险识别过程将确认哪些风险影响到项目并且记录这些风险的特征。参与风险识别的人员应包括项目经理、项目团队成员、风险管理团队、风险管理领域的专家、客户、最终用户等。在风险管理过程中，鼓励所有的项目成员来进行风险识别。

3）风险识别方法与技术

风险识别是任何风险管理活动的起点，项目风险识别的研究方法大致可以有如下几种：信息收集技术、检查表、穷举法、假设法、任务分解识别法、图表法分析和财务报表法。

(1) 常用的信息收集技术包括头脑风暴、Delphi法、面谈、问卷调查、SWOT分析法和情景分析法。

(2) 检查表是将项目可能发生的许多潜在风险列于一张表上，供识别人员进行检查核

对，以判别项目中是否存在表中所列或类似的风险。

(3) 穷举法风险识别清单是建立在历史信息和过去相似项目的经验之上的。Barki 等通过总结列出了 35 项风险变量，Jones 描述了 60 项最常见的风险因素。

(4) 分析假设的合理性将为风险识别提供方便。

(5) 任务分解识别法就是将项目管理过程中遇到的复杂的难于理解的问题分解成比较简单的容易被识别的问题，以易于发现风险。

(6) 图表分析包括因果效果图、系统流程图和效果图等，通过图表的方式有助于直观分析对比风险。

(7) 通过分析资产负债表、营业报表，以及财务记录，项目风险经理就能识别本企业或项目当前的所有财产、责任和人身损失风险。将这些报表和财务预测、经费预算联系起来，有助于发现未来的风险。

4) 风险识别结果

当识别了项目风险之后，进行分类和合并，形成风险列表，其中包括风险名称和简要说明。

2. 风险分析

风险评估又称风险预测，是对已识别的风险从风险发生的概率与影响程度两个方面来进行估计和评价，得出风险分析表，包括：①风险名称，即所有已识别的风险；②风险发生的可能性或概率；③风险发生会产生的影响。

风险评估的方法有很多种，概括起来可分为三大类：定量的风险评估方法、定性的风险评估方法、定性与定量相结合的评估方法。

(1) 定量的风险评估方法。定量的风险评估方法是指运用数量指标来对风险进行评估，典型的定量分析方法有因子分析法、聚类分析法、时序模型、回归模型、风险图法和决策树法等。定量的评估方法的优点是用直观的数据来表述评估的结果，看起来一目了然，而且比较客观。定量分析方法的采用，可以使研究结果更科学、更严密、更深刻。

(2) 定性的风险评估方法。定性的风险评估方法主要是指依据研究者的知识、经验、历史教训、政策走向及特殊实例等非量化资料对系统风险状况做出判断的过程。典型的定性分析方法有因素分析法、逻辑分析法、历史比较法和 Delphi 法。定性风险评估方法的优点是可挖掘出一些蕴藏很深的思想，对评估者本身的要求很高。

(3) 定性与定量风险评估。系统风险评估是一个复杂的过程，需要考虑的因素很多。一般认为定量分析是定性分析的基础和前提，定性分析应建立在定量分析的基础上才能揭示客观事物的内在规律。在复杂的信息系统风险评估过程中，应该将这两种方法融合起来，采用综合的评估方法。

风险评估的常用技术有综合评价法、风险因子计算、PERT(Program Evaluation and Review Technique，计划评审技术)估计、风险评审技术 VERT(Venture Evaluation and Review Technique，风险评审技术)、GERT 估算、失效模式、影响与致命度分析(FMECA)、风险决策树分析、故障树分析法(FTA)、事件树分析法(ETA)、蒙特卡洛方法(属于概率统计方法范畴)、决策树分析、风险模拟和专家判断等。

在进行风险模型、评估标准、评估方法研究的同时，各大公司或研究人员也相应推出评

估工具来辅助进行评估。例如，SAFESuite 套件、WebTrends Security Analyzer 套件、Cobra、CC tools、MSAT、ASSET 等。这些工具有的是通过技术手段，如漏洞扫描、入侵检测等来维护信息系统的安全；有的是依据评估标准而开发的，如 Cobra。

3. 风险优先级评定

风险分析表中风险发生会产生的影响，按优先级进行表示，通常表示为 1(低)～4(高)之间的序数范围。

10.4.3 风险控制

风险被识别和评估之后，Web 工程项目团队必须制定控制风险的计划。计划包括选择针对不同风险所要采取的风险应对措施，持续对风险进行监控，然后尽可能降低风险。

风险应对需要有应对计划和应对储备。应对计划是指当一项已识别的风险事件发生时，项目团队将采取的预先处理的措施。应对储备是为了应付项目可能发生风险所持有的预备资金，可用来转移成本和进度风险。风险监控包括风险监督和风险控制，是指跟踪已识别出的风险。风险降低是设法避开或转移风险，或在风险发生后实施相应计划以把损失降低最低。风险监控和风险降低是一个循环迭代的过程。

1) 需求风险控制措施

(1) 前期的需求讨论要详细、充分，范围要明确，功能描述要清楚。

(2) 需求文档中要有演示版。对于 Web 项目，图片比文字更能说明问题。

(3) 找出项目中需求的决策者(通常会是产品经理、相关职能主管、客服)，所有的需求要经过他们的认可。

(4) 客户在项目过程中的全程参与有助于降低此类风险，客户参与需求讨论、需求确认、用例确认、测试阶段的客户验收等环节。

(5) 发生需求变更时，严格按照需求变更流程执行。

2) 技术风险降低措施

(1) 在项目开始前的技术评估阶段，明确技术难点，提前安排人员进行攻克。如果在可预期的时间内无法解决，可以要求需求方变更需求。

(2) 可以参考同行经验。针对新技术，由于没有经验可借鉴，因此在探索过程中要充分利用互联网，通过搜索同行经验，往往事半功倍。要充分利用世界日益平坦化的优势，对于不能尽快解决的问题，可以先放一下，可能过不了多久，网上就有相类似问题的解决方案，到时候就可以借鉴这些同行们的宝贵的经验来解决实际的问题。

3) 质量风险降低方式

(1) 制定尽可能合适的开发时间计划，以减少不合理的开发时间对开发质量的影响。

(2) 有一套严格可行的代码规范，编码时严格遵守，代码复查时严格考核。

(3) 在编码前，开发人员要对所使用的框架熟练掌握。

(4) 一份好的系统设计文档对指导开发非常重要。

(5) 代码走查、会议评审和同行专家评审来确保软件质量。

(6) 质量审查工具软件的使用。充分利用质量审查的工具软件，也有利于提高代码质量。例如，在 Eclipse 开发环境中，可以集成 Findbug、Checkstyle、PMD 插件等工具检查代

码编写质量。

4）资源风险降低方式

需要在项目计划制定时提前确认资源，并在项目进行过程中不断沟通协调。

5）性能风险降低方式

（1）在系统设计时，应做好前期性能规划，对可能出现性能问题的环节做到充足的估计。

（2）在开发过程中，要重视性能测试和压力测试，尽可能模拟实际使用环境，搭建测试平台，并使用性能测试工具以及相关分析工具。

6）安全性风险降低方式

（1）制定应用安全的工具和标准。

（2）完整的应用安全测试、代码安全开发和评审的书籍。

（3）标准的安全控制和库。

（4）尖端技术研究。

（5）全球化沟通。

（6）邮件列表等。

10.5 Web项目配置管理

由于Web应用开发规模日益庞大，结构日益复杂，人员流动快速，在软件开发过程中，经常会出现下述影响开发计划和质量的情况。

（1）无法对用户需求进行有效地管理和追踪。

（2）产品升级和维护所必需的程序和文档少而乱。

（3）很多版本之间并行开发，使并行开发小组之间沟通面临挑战，导致一致性和集成问题。

（4）开发过程中项目组成员流动大，造成项目后继人员接收前人工作困难。

（5）上线时间紧，导致部分未经充分测试的软件加入到产品中。

在Web应用开发和使用环境中，如不对以上这些情况采取措施，将会对开发造成巨大破坏，情况严重者甚至直接导致项目失败。因此，必须使用有效的高效的软件配置管理，确定基线，并跟踪这些基线的变更，使改进变更易于被适应，并减少当变化必须发生时所需花费的工作量。

10.5.1 Web配置管理的内容

Web项目配置管理是一项很重要的活动，包含一组行为来控制和管理变更，一般分为配置识别、变更控制、状态报告和配置审计，不过这种分解方式的灵活性和严格性并不非常适合Web应用开发项目。为了清晰性、可扩展性和适应性，本节针对Web应用开发项目采用如下分解方式：版本控制、文档控制、变更管理、构建管理和发布控制。这种分类方式更加针对完成Web应用所提供的功能，同时控制基线和变更，强调更快地执行有形的任务，而非生成文档。

1）版本控制

针对最终用户而言，只有一个可用的Web应用版本，但是也可能存在其他版本。老版本可能出于历史原因被存档，包含了新的美学设计、内容和功能（沿着一些新的导航路径）的新版本可能正在开发中。

版本控制是对系统不同版本进行标识和跟踪的过程，在有版本开发完成后就设立，涉及项目组所有成员对版本的各种操作的控制，包括检入（Check In）和检出（Check Out）控制、版本的分支和合并、版本的历史记录和版本的发行。版本控制作为配置管理系统的核心，它所控制的对象是开发过程中涉及的所有文件系统对象，包括文件、目录和链接。

Web应用版本控制工具应支持如下操作：

- 为任意类型的文件设置版本。
- deltas的前翻和倒转。
- 文件压缩。
- 分支。
- 特定情况下自动创建分支。
- 版本标签。
- 视觉差别。
- 视觉合并。
- 并行开发。
- 支持项目和文件。
- 支持子项目。
- 不受限的目录层次。
- 层次中的递归命令。
- 追踪项目结构的变化。
- 多个应用间的项目/文件共享。

Web应用必须考虑基线，不仅仅是源代码，还包括可执行码、图形（GIF和JPEG文件）、数据文件以及创建完整应用所需的任意类型的数字物品。如今，在很多情况下，内容管理仅仅是生产中的Web应用的版本控制的新的名字而已，但是所使用的工具从存储库到产品服务器都进行了控制和受控项交付方面的优化。Web应用还必须特别关注核对合法使用权利，同时把版本控制收到的图像进行存档。对于很有创造性和奇思妙想的设计者来说，可能忽略了版权的授权。因此，在任何图像被接受进入版本控制的同时，获取版权授权的元数据，而且，所有的HTML代码中都插入版权保护声明，这样Web用户在剪切和粘贴代码时就能够阻止未授权的使用。

2）文档控制

文档控制把过程和工具进行结合以管理和保存开发过程模型中的所有识别的文档的版本，在开发模型的文档被批准之后就开始执行，涉及开发团队所有成员。文档管理提供了一种保存和检索开发模型中使用的文档的方法，这样就能维护决策的软件基线的可追溯性，同时有了可用于诉讼和协商的基础。

Web应用文档控制工具，应支持如下操作：

- deltas的前翻和倒转。

- 为任意类型的文件设置版本。
- 文件压缩。
- 分支。
- 特定情况下自动创建分支。
- 版本标签。
- 视觉差别。
- 视觉合并。
- 并行开发。
- 支持项目的同时支持文件。
- 支持多线程。
- 不受限的目录层次。
- 层次结构中的可递归命令。
- 追踪项目结构的变化。
- 多个应用系统间共享文件。

将文档管理从版本管理中分离出来是为了提供更大的用户组访问,支持更好的成本效益,提供高效访问知识库的方式。另外,Web应用的文档控制系统必须能够在文档中内嵌超链接,关注全球化,即对Web页面中不同文字和编码标准的支持。

3）变更管理

变更管理是系统的初始基线之后启动对系统配置的所有批准的变更的提议、说明、评估、协调、批准或不批准、实现,以控制变更可能导致的混乱,使系统的变更能在成本和时间方面更加高效和可控,涉及包括客户在内的开发团队中所有成员。

变更可以分为两类：微粒级和粗粒级。微粒级,是每个开发人员每天都要对组件或一系列组件进行的变更,例如,BUG修复或小的变更请求。粗粒级,是新增的主要产品功能。

在实际操作的过程中还会有两种情况会对变更管理产生影响。一是变更流程的制定虽然得到了开发人员的认可,但执行往往不利；二是如果项目时间紧,变更多,那么变更流程执行的效果往往不好。

变更管理通常通过变更控制追踪系统和版本控制系统两个配置管理工具来完成。变更控制系统必须连接到版本控制系统。变更控制系统对变更请求的初次出现进行登记,将请求进行分析检查,之后变更请求等待配置管理委员会的处理决定。然后系统把请求分发给指定的实施者,记录实施的描述,然后把请求送回配置管理委员会进行实施的批准和发布基线的合并。所有的变更都必须具有到达版本控制系统中获取特定配置项的链接。

Web应用开发对时间要求很高,基于纸质和大量会议的时间密集型变更管理系统是几乎不可能的。Web应用开发成功的变更管理的两个因素是：使用能够在系统批准的决策者做出行动决策时自动通知和转发的工具,使变更过程自动化；授权项目经理进行复审。在项目经理批准变更的过程中,由项目经理来征集和获取其他人的意见。

4）构建管理

构建管理是执行或验证所有贯穿于受控文档的配置基线的构建的功能,在任何配置基线产生时都会被执行,提供了所有配置基线的可跟踪性和可重复性,这样交付给客户的产品就能够跟踪到源文档。构建管理的执行可以是任何指定的项目团队成员,但是这些构建如

何完成的记录以及详细说明完成构建的技术步骤的文档化过程，必须由配置管理分析员来检查和存档。

构建管理可以通过一个与版本控制工具交互的自动化实用工具来实施，也可以通过一个可以获取命令和引用的记录脚本来实施，这些引用对产品中分离执行的构建来说非常必要。基于基线的版本控制脚本也置于档案文件内，并链接到相应基线。

构建管理对于传统的软件配置管理人员来说是一个陌生的主题，可是，由于Web应用的复杂性，它却是Web应用要处理的一个主要挑战。对于理论模型来说，一个简单的版本描述文档就足以应对后面的配置审计。

对Web应用而言，不仅所有的软硬件都必须被详细列出以明确识别，还必须为成功构建产品服务器而记录软硬件设置和安装的顺序。记录者还需创建"配置说明"文档，明确记录所有硬件项标识、软件项标识、环境设置、进入系统的所有命令以及这些行为的适当执行顺序。该文档大大超出了常见版本描述文档中所描述的信息。另外，该文档通过从头设置测试服务器进行确认，然后才用于设置产品服务器。

5）发布控制

发布控制是收集、记录和传输所有的交付品到运行地。所有的可交付品都必须执行配置审计（这里是指为预期交付给外部各方的软件而进行的产品审计），以确保配置被精确记录，构建程序和系统文档的可跟踪性被验证，以确保所有可交付品的可追踪性。对Web应用而言，配置审计必须依照"配置说明"来进行构建，以确认"配置说明"，从头开始设置产品服务器。该过程在使用预发布服务器(Staging Server)或测试实验室等受控的环境下进行，通过使用版本控制实用工具记录所有外部交付品的配置来实现。

对于复杂的Web应用而言，发布前确保Web应用的质量至关重要，需要较严格划分的物理环境，将开发服务器、测试服务器、预发布服务器和产品服务器划分为不同的服务器。以使Web应用在受控的环境下进行迁移，实现21×7运行且不死机的时间可靠性，保证迁移满足预定义的需求和基线，满足系统文档要求，提供综合风险降低策略，最终提供可靠的运行系统。

因为Web应用的发布的特性，所以将发布控制作为配置管理的一项活动。Web应用成为产品后，所有的变更都必须被跟踪，而且可以跟踪到受控项，这样才能有效地进行内容管理。而且，因为必须快速排除出现的缺陷或功能的增强，项目经理经常具有决定发布的权利。

10.5.2 配置管理的实施

实施配置管理，一般的步骤和需要考虑的问题如下。

（1）规划、调整网络开发环境。一个规划良好的开发环境，是实施配置管理系统的前提，主要考虑网络的带宽、拓扑结构，服务器的选择、命名规范，存储区的定位，开发人员及组的命名规约等问题。

（2）设计配置管理库。配置库一般包括开发库、基线库、产品库，其中开发库存储项目的所有工作产品中间结果，基线库存储项目的所有基线，产品库存储所有对用户发布的版本。配置库作为项目组内成员今后工作的平台，前期的准备是非常重要的。配置库建立包

括环境的搭建和有关配置库使用制度的规定，之后需要及时对员工提供工作指导和配置库使用的操作培训。

(3) 定义配置管理系统的角色。在此阶段，需要确定与配置管理相关的所有角色，包括他们的相应的活动。在开发过程中，一个开发人员可能兼任多种角色，但一项任务在同一时刻只能由一个角色来执行。一般配置管理中的角色及其职责如表 10.3 所示。

表 10.3 配置管理的角色及其职责

SCM 角色	职责
项目经理(PM)	制定和修改项目的组织结构和配置管理策略；批准、发布配置管理计划；决定项目起始基线和开发里程碑；接受并审阅配置控制委员会的报告
配置管理委员会(CCB)	制定开发子系统；制定访问控制；制定常用策略；建立、更改基线的设置，审核变更申请；根据配置管理员的报告决定相应的对策
开发经理	定义开发的子系统；定义访问控制；制定一般策略；制定重要集成里程碑事件
配置管理员(CMO)	配置管理工具的日常管理与维护；提交配置管理计划；各配置项的管理与维护；执行版本控制和变更控制方案；完成配置审计并提交报告；对开发人员进行相关的培训；识别软件开发过程中存在的问题并拟就解决方案
集成人员(SIO)	集成修改；构建系统；完成对版本的日常维护；建立外部发布版本
开发人员(DEV)	创建开发视图；处理变更；根据确定的配置管理计划和相关规定，提交配置项和基线；负责软件集成和版本生成；按照软件配置管理工具的使用模型来完成开发任务
构建人员	构建可选版本；创建发布介质
测试人员	集成测试；系统测试
质量保证人员	负责配置审核并提交报告；对配置审核中发现的不符合项，要求相关负责人进行纠正

10.5.3 配置管理的工具

Web应用项目团队需要多人协同工作。如果代码管理混乱，会使代码的冲突解决困难，代码集成也会有比较深层次的缺陷，而且对于经常需要发布新版本的 Web 应用，也非常麻烦，特别是对大型项目和异地协同开发而言，问题更加严重。因此，采用合适的版本控制工具，以有效解决版本引起的各种问题，让开发人员能把更多的精力花费在开发上，加速过程决策。

1. 常用工具

1) PHProjckt

PHProjckt 是一个模块化的协同办公系统，用于共享信息和文档。它包括的组件有团队日历、Time Card 系统、项目管理、请求跟踪、文档管理、通讯录管理、E-mail 客户端、论坛、聊天、记事本、共享书签、待办事项目列表和投票系统等。

2) CCC、SCCS、RCS

CCC 是 Platinum 公司的一个基于团队开发的提供以过程驱动为基础的配置管理工具，包含版本管理、过程控制等功能。它可以在异构的平台，远程分布的开发团队以及并行开发活动的情况下保持工作的协调和同步。它还可以有效跟踪复杂的企业级开发的各种变化

(变更)的差异,从而使企业可以在预定的交付期限内提交高质量的应用系统。

3) Rational ClearCase

ClearCase 是 Rational 公司开发的配置管理工具,是现在应用面最广的企业级、跨平台的配置管理工具之一。ClearCase 提供比较全面的配置管理支持,包括版本控制、工作空间管理、构建管理等,开发人员无须针对其而改变现有的环境、工具和工作方式。

4) Hansky Firefly

Firefly 是 Hansky 公司软件开发管理套件中的重要一员,它是一个功能完善、运行速度极快的软件配置管理系统,可以支持不同的操作系统和多种集成开发环境,因此它能在整个企业中的不同团队、不同项目中得以应用。它可以轻松管理、维护整个企业的软件资产,包括程序代码和相关文档。Firefly 基于真正的客户机/服务器体系结构,不依赖于任何特殊的网络文件系统,可以平滑地运行在不同的 LAN、WAN 环境中。

5) CVS

CVS(Concurrent Versions System)是一款开放源代码软件,它使用简单,功能强大,支持跨平台,支持并发版本控制,在全球中小型软件企业中得到了广泛使用,是目前最流行的面向软件开发人员的源代码版本管理解决方案。它可用于各种平台,包括 Linux、UNIX 和 Windows NT/2000/XP 等。

6) Merant PVCS

PVCS 是 Merant 公司开发的一款软件配置管理工具,它通过使用其图形界面或类似 SCCS 的命令,能够基本满足小型项目开发的配置管理需求。PVCS 虽然功能上也基本能够满足需求,但其不足之处是性能表现一直较差。

7) VSS

VSS(Visual Source Safe)是微软公司为 Visual Studio 配套开发的一个小型的配置管理工具。VSS 的优点在于其与 Visual Studio 实现了无缝集成,使用简单。但其缺点也是十分明显的,只支持 Windows 平台,不能在 Linux、UNIX 等平台下运行,不支持并行开发,通过 Check out - Modify - Check in 的管理方式,一个时间只允许一个人修改代码,而且速度慢、伸缩性差,不支持异地开发。

2. 接口

对于 Web 应用所采用的配置管理工具而言,有必要提供如下几个系统接口,要么和系统集成为一个整体,要么提供来回传输信息的 API 接口。

(1) 通过把变更管理工具与组织的电子邮件系统进行链接,就能够自动记录和通知要求的动作或决策,能够将变更请求传输到下一个未来动作的决策点。

(2) 由于需要快速流畅地接收和响应客户的缺陷修复或强化的请求,在客户支持自动化软件和软件配置管理工具之间要有 API,来接收客户请求,并传输到软件配置管理系统。在对请求的决策或进展过程中的任何时候,客户支持代表都可以查看实时状态。

(3) 内容管理交付系统必须使用 API 与软件配置管理系统进行链接,以进行 Web 应用的必要调整,这对于支持必须要显示和操纵的新数据和图像来说非常必要。

(4) Web 应用必须在软件配置系统中可控,以避免操作系统和 Web 应用中集成的第三方软件之间的冲突。

10.6 总结与展望

Web项目管理是为了使Web应用能够按照预定的成本、进度和质量顺利完成，而对成本、人员、进度、质量和风险等进行分析和管理的活动。对于以项目为基本运作单位的开发团队而言，主要目的是让每个项目都能保质保量完成，达到双赢的效果。Web应用的自身特性，使得Web项目管理面临众多挑战。而相应地，Web项目进度管理、Web项目成本管理、Web项目人员管理、Web项目风险管理、Web项目配置管理等方面都有其自身特性，以及特有的方法和技术等，强调实用原则。

敏捷开发方法已经成为Web应用项目的主要开发方法，Web应用项目管理也体现出了更多敏捷的特点。因而，在Web项目管理中也开始了大量关于敏捷和刚性的研讨。结合Web应用项目管理的特点，将敏捷的理念运用到Web应用项目管理中，采用敏捷项目管理的思想，使得Web项目管理兼具敏捷和刚性的特点，以适应Web应用的新变化。

第11章 Web应用的性能和可用性

性能和可用性对一个Web应用而言是非常重要的质量指标。判断Web应用成功与否的一个重要评估标准是Web应用的性能，Web应用性能的好坏直接影响到用户的满意度。而Web应用的可用性又对Web应用的质量起着至关重要的作用。

用户一般不是计算机专业人士，他们使用计算机的目的只是为了满足自己的工作、娱乐和交流等需要。因此，Web应用只有能够提高用户的工作效率或是方便他们的日常生活才是有用的，而不是留给他们很长的等待时间。如果Web应用的性能达不到用户的期望，很难想象他们还有耐心等待着去使用它。Web应用首先必须满足用户的基本使用要求，只有达到了用户的可用性要求，才能为用户所接受。

目前，Web应用环境的快速发展，对性能的度量与提升、可用性工程提出了新的需求和挑战，需要使普通Web应用能够综合使用度量、分析、改进及其相互作用自动化，能够适应移动互联网环境，激烈的社会竞争也使Web应用需要给各种具有普通使用方式有障碍的人士提供可访问性支持。

11.1 Web应用性能

用户一般很难接受响应缓慢的Web应用，绝大多数用户都不会执著地等待响应迟钝、加载缓慢、等待时间很长的Web应用而不去尝试访问别的Web应用。如果一个Web应用的性能太差，那么用户的满意度会极度降低，进而对公司的经济利益甚至声誉造成损失。

11.1.1 Web应用性能分析

评估一个Web应用的性能如何，不是Web应用开发人员说了算，通常需要站在用户的角度来看待这个问题。通过访问该Web应用的一系列Web页面，体验页面设计、等待时间、下载速度等指标，进而评价其性能的好坏，并相应进行提升。

1. 用户访问过程

当用户访问一个Web应用的时候，用户首先输入Web应用的地址，通常为其URL，等待Web服务器的响应。浏览器获得了用户希望访问该地址的意图，便向Web服务器发起一系列的请求，这些请求不仅包括对Web页面的请求，还包括对Web页面中许许多多组件的请求，比如图片、层叠样式表、脚本、内嵌页面等，甚至还包括从应用服务器端动态生成页

面。接下来的一段时间,浏览器等待服务器的响应以及返回的数据。待浏览器获得所有的返回数据后,经过客户端本地的计算和渲染,最终一幅完整的 Web 页面才呈现在用户的眼前。

在客户端向服务器发送请求到服务器响应客户端请求的这个过程中,大概经历了以下三个过程的时间。

① 数据在网络上传输的时间。数据在网络上传输的时间总的来说包括两部分,即客户端主机发出的请求数据经过网络到达 Web 服务器的时间,以及 Web 服务器的回应数据经过网络回到客户端主机的时间。这两部分时间都可以视为某一大小的数据从某主机开始发送一直到另一端主机全部接收所消耗的总时间,称它为响应时间,它的决定因素主要包括发送的数据量和网络带宽。

② 服务器处理请求并生成回应数据的时间。Web 服务器处理请求并生成回应数据的时间主要消耗在服务器端,包括非常多的环节,一般用另一个指标来衡量这部分时间,即每秒处理请求数,也称吞吐率。注意这里的吞吐率不是指单位时间处理的数据量,而是请求数。影响服务器吞吐率的因素非常多,比如服务器的并发策略、I/O 模型、I/O 性能、CPU 核数等,当然也包括应用程序本身的逻辑复杂度等。

③ 浏览器本地计算和渲染的时间。浏览器本地计算和渲染的时间自然消耗在浏览器端,它依赖的因素包括浏览器采用的并发策略、样式渲染方式、脚本解释器的性能、页面大小、页面组件的数量、页面组件缓存状况、页面组件域名分布以及域名 DNS 解析等,并且其中一些因素随着各厂商浏览器版本(最主要的是浏览器引擎和脚本引擎)的不同而略有变化。

可见,一个 Web 页面包含了若干个请求,每个请求都或多或少地涉及以上这些过程,假如有一处关键环节的处理速度不尽如人意,那么整体的速度便会受到影响。因此,在进行 Web 应用性能提升时,应该综合从这几个方面考虑。

2. 性能指标

在评估 Web 性能中,会用到很多特有术语,每个都有其特殊的意义,以下列举出常用的并且重要的术语,并对它们进行解释。

(1) 连接时间。客户端和 Web 服务器间建立连接所需要的时间,通常以秒为单位进行计算。

(2) 发送时间。从客户端向 Web 服务器发送数据需要的时间,通常以秒为单位进行计算。

(3) 接收时间。从 Web 服务器向客户端发送响应数据所需要的时间,通常以秒为单位进行计算。

(4) 处理时间。Web 服务器响应客户端请求所需要的时间,通常以秒为单位进行计算。

(5) 响应时间。完成某个特定事务所需要的时间,通常以秒为单位进行计算。

(6) 并发用户。并发一般分为两种情况。一种是严格意义的并发,即所有的用户在同一时刻做同一件事情或者是操作,这种操作一般指同一类型的业务。例如,在信用卡审批业务中,一定数目的用户在同一时刻对已经完成的审批业务进行提交(操作的不是同一条记录)。这一类里面也有一种特例,即所有的用户进行完全一样的操作。另一种并发是广义范

围的并发，这种并发与前一种并发的区别是，尽管多个用户对系统发出了请求或进行了操作，但是这些请求或者操作可以是相同的，也可以是不同的。对整个系统而言，如果有很多用户同时对系统进行操作，也属于并发的范畴。并发用户的数量越多，Web应用的负载越大，对Web服务器和Web应用的要求也就越高。

(7) 请求响应时间。指的是客户端发出请求到得到相应的整个过程的时间。在某些工具中，请求响应时间可以用"TTLB"(Time To Last Byte)，意思是从发起一个请求开始到客户端收到最后一个字节的响应所消耗的时间，其单位一般是"秒"或者"毫秒"。

(8) 事务响应时间。事务可能是由一系列请求组成的，事务的响应时间主要是针对用户而言的，属于宏观上的概念，是为了向用户说明业务响应时间而提出的。例如，跨行取款事务的响应时间就是由一系列的请求组成的。事务响应时间和后面的业务吞吐率都是直接衡量系统性能的参数。

(9) 吞吐量。在给定时间范围内网络上传输的数据量的总和。找出能保持发送帧数与接收帧数相等的发送速率最大值。此时发送或者接收的帧数就是吞吐量。

(10) 吞吐率(Throughput)。吞吐量/传输时间，就是吞吐率，即单位时间内网络上传输的数据量，也可以指单位时间内处理的客户端请求数量。它是衡量网络性能的重要指标。通常情况下，吞吐率用"请求数/秒"或者"页面数/秒"来衡量。

(11) 每秒事务处理量(Transaction Per Second，TPS)。每秒钟系统能处理的交易或者事务(如每秒Head、Get、Post请求)的数量，它是衡量系统处理能力的重要指标。

(12) 点击率(Hit Per Second)。每秒钟用户向Web服务器提交的HTTP请求数，是Web应用特有的一个指标。Web应用是"请求—响应"模式，用户发出一次申请，服务器就要处理一次，所以点击是Web应用能够处理的交易的最小单位。如果把每次点击定义为一个交易，点击率和TPS就是一个概念。可以看出：点击率越大，对服务器的压力越大。点击率只是一个性能参考指标，重要的是分析点击时产生的影响。需要注意的是，这里的点击不是指鼠标的一次"单击"操作，这是因为在一次"单击"操作中，客户端可能向服务器发出多个HTTP请求。

(13) 资源利用率。资源利用率是指对不同系统资源的使用程度，例如服务器的CPU占用率、磁盘利用率等。资源利用率是分析性能指标进而改善性能的重要指标，因此是Web性能测试工作的重点。负载均衡(Load Balance)是提高资源利用率的一种有效方法。

11.1.2 Web应用性能提升策略

要提高Web应用的性能，可以采用多种策略，将这些策略综合使用，才能发挥最好的效果。以下列举的是一些常用的提高Web应用性能的方法。

1) 增加带宽

网络速度是影响Web应用性能的重要因素之一，网络速度太慢时，Web应用的响应时间自然变长，性能自然就很低了。所以当Web应用的Web页面或组件的下载速度变慢时，可以采用增加服务器带宽的方法来提高加载、下载速度，进而提高Web应用的性能。

2) 减少Web页面中的HTTP请求

仅有10%～20%的用户终端响应时间涉及对被请求的HTML文档进行检索，剩下的80%～90%的时间用于对HTML文档中所引用的所有组件(图片、脚本、样式表等)进行

HTTP 请求。因此,一种显而易见提高 Web 应用性能的方法就是减少组件数,也就是减少 HTTP 请求。常用的可以有效降低 HTTP 请求数的技术有图像映射、对脚本与样式表分别进行合并等。

3) 加快服务器脚本计算速度

大多数涉及性能问题的 Web 应用都会使用各种各样的服务器端脚本语言,比如主流的 PHP、Ruby、Python、ASP. NET、JSP 等,这些脚本语言用来编写动态内容或者后台运行的小程序,已经成为几乎所有 Web 应用的首选。

用脚本语言编写的程序文件需要通过相应的脚本解释器进行解释后生成中间代码,然后依托在解释器的运行环境中运行。所以生成中间代码的这部分时间又成为大家为获取性能提升而瞄准的一个目标,比如解释器对某个脚本程序第一次解释的时候,将中间代码缓存起来,以供下次直接使用。

4) 采用缓存技术

缓存是一种介于架构与代码级别优化之间的重要优化技术。网络缓存技术的目的是减少网络中冗余数据的重复传输,使之最小化,将广域传输转为本地或就近访问。互联网上传递的内容,大部分为重复的 Web 或 FTP 数据,缓存服务器及应用缓存技术的网络设备,可大大优化数据链路性能,消除数据峰值访问造成的结点设备阻塞。缓存服务器具有缓存功能,所以,大部分页面对象,在有效期(TTL)内,对于重复的访问,不必从原始 Web 应用重新传送文件实体。缓存服务器不仅能提高响应速度,节约带宽,还能有效减轻源服务器的负荷。

根据缓存所处的位置不同,可以把缓存技术分为三类:客户端的缓存技术、基于代理服务器的缓存技术和服务器端的缓存技术。

5) 使用代理服务器

代理服务器是介于客户端和 Web 服务器之间的另一台服务器。当用户访问 Web 应用时,浏览器不是直接到 Web 服务器去取回 Web 页面而是向代理服务器发出请求,信号会先送到代理服务器,由代理服务器来取回浏览器所需要的信息并传送给客户端的浏览器。

使用代理服务器,可以提高访问速度,对经常访问的地址创建缓冲区,大大提高热门 Web 应用的访问效率。通常代理服务器都设置较大的硬盘缓冲区(可能高达几个 GB 或更大),当有外界的信息通过时,同时也将其保存到缓冲区中,当其他用户再访问相同的信息时,则直接由缓冲区中取出信息,传给用户,以提高访问速度。

6) 动态内容静态化

在动态内容缓存技术的实现机制中,虽然避免了大量的重复计算,但是每次还都需要调用动态脚本解释器来判断缓存是否过期以及读取缓存,这会消耗不少时间。直接让浏览器访问这些动态内容的缓存效果会更好,在这种情况下缓存成为直接暴露给前端的 HTML 页面,而整个缓存控制机制也发生了根本的变化,一般称它为静态化。

7) 页面容量优化

Web 页面设计人员的精巧设计将会显著减少通过网络传递 HTTP 请求和响应所花费的时间。终端用户的带宽速率,互联网服务提供商与交换结点的邻近程度,以及其他因素都超出了开发团队的控制范围,但是,仍然有着可以施加影响的因素影响着响应时间。如果对于一个 HTTP 请求,给予更少的响应内容,那么传输时间将因为更少的包在服务器和客户

端之间传递而减少。这将对较低的带宽速度更加有效。对脚本和样式表进行压缩是减少Web页面负担最简单的技术,并且会产生最大的作用。有文件级别和代码级别两种压缩方式。文件级别的压缩用于邮件信息和FTP站点中缩小文件大小的文件压缩,它同样可以用于向浏览器分发被压缩的Web页面。代码级别的压缩是从代码中移去不必要的字符代码,进而提高Web应用性能。

8）页面组件分离

如果由同一台物理服务器或者同一种并发策略的Web服务器软件来统一提供服务,那势必会造成计算资源的浪费以及并发策略的低效。所以,分离带来的好处是显而易见的,那就是可以根据不同组件的需求,如下载量、文件大小、对服务器各种资源的需求等,有针对性地采用不同的并发策略,并且提供最佳的物理资源。

9）提升Web服务器性能

运用更快速的磁盘和更好的网络存取机制,能明显改进网络访问速度。这类方案的核心是设法减轻Web服务器CPU的负荷,使其从繁琐的网络协议处理中"解脱"出来,而集中于页面处理和服务提供。此外,在不改变硬件资源和应用的前提下,通过调整服务器软件的配置参数也可以评价和优化系统的性能。

10）使用负载均衡

负载均衡是实现多台服务器协同工作和并行处理的手段,其核心思想是采用增加同时工作的主机数量的方法,根据当前服务器的负载情况,将到达的大量请求报文分配到不同的服务器处理,从而减轻单个服务器的处理负担,可以极大地提高服务器性能,使资源得到最大效率的利用。

负载均衡是大型Web应用解决高负荷访问和大量并发请求所常用的一种解决方案。一般来说,负载均衡技术包括DNS负载均衡、代理服务器负载均衡、地址转换网关负载均衡、协议内部支持负载均衡、NAT负载均衡以及混合型负载均衡。

11）优化数据库

对于使用数据库的Web应用来说,优化数据库能够大大地提高Web应用的性能。往往一些性能问题可能都发生在表现不佳的数据访问层面,这来源于不合理的应用程序数据访问组件设计、不合理的数据库表结构设计以及对于数据库内部构造缺乏深入的了解。

良好的数据库设计可以大大改善访问效率,一个好的数据库本身提供了大量的优化设计措施以提高数据库的访问性能。常用的数据库设计方法有建立索引、关键字、簇和优化表结构两种形式。

12）优化应用程序

根据统计,对网络、硬件、操作系统、数据库参数进行优化所获得的性能提升,全部加起来只占数据库系统性能提升的约40%,其余60%系统性能提升来自于对应用程序的优化。应用程序的优化通常可分为两个方面：源代码优化和SQL语句优化。由于涉及到对程序逻辑的改变,源代码的优化在时间成本和风险上的代价都很高。

13）使用CDN

分布式资源体系结构就是CDN(Content Delivery Network,资源分发网络),是指一份资源可以有多份复制件,而这些复制件分布在Internet上的不同地方。其目的在于针对用户所在的位置选取从最合适(如最近或最空闲)的地点提供服务。

分布式资源体系结构需要解决两个主要问题：①如何针对用户选取最佳的资源服务器；②如何在资源服务器之间传送资源或资源的索引。目前后者的一种解决方案是每个资源服务器维护一份独立的资源复制件，该资源的权威服务器定期对镜像服务器进行更新；另一种方案是形成资源交换网络，同路由器之间交换路由信息一样，交换网络中间的结点交换的是资源对象的信息。每个结点保存的不是资源的复制件，而是可以最快获取该资源的路由，用户对该资源的请求会被转发到该处。

14）采用镜像分担流量

镜像站点是指将某些 Web 应用内容映射到其他 Web 应用，以分担访问流量。镜像是大型 Web 应用常采用的提高性能和数据安全性的方式，镜像的技术可以解决不同网络接入商和地域带来的用户访问速度差异，访问国外站点的国内镜像，可以提高访问速度，减少国际流量。

11.2 Web 应用可用性

可用性（Usability）也称为以用户为中心的设计（User Centered Design，UCD）或人性因素，是指能够让客户快速达成其目的，并且让用户满意的能力。ISO 对可用性的定义“Usability is the effectiveness, efficiency and satisfaction with which a specified set of users can achieve a specified set of tasks in a particular environment.”给出可用性的核心是用户在特定环境中完成特定任务的效益、效率和用户满意。根据 Jakob Nielsen 的定义，可用性关注用户界面的多个方面，其中包括用户满意。Whitney Quesenbery 更加强调满足用户需求，增强用户体验（excellent user experience）。

可用性的核心在于以用户为本，快速、明确、简单、方便且可信。Web 应用可用性是关于 Web 应用是否容易使用的一种质量属性，更准确地说，是用户是否能够快速地使用 Web 应用的能力，即使用效率如何，使用方法是否简单易记，使用时是否容易出错，以及用户是否喜欢使用它。简单来说，就是有好的用户体验。如果用户无法使用或不愿使用某个功能，那么该功能还不如不存在。

Web 应用的可用性相比传统软件产品而言，由于其全球性和基于 Internet 的特点，和传统软件的可用性有着很多差别。

（1）不再以盒装软件方式发售，WYSIWYG 给 Web 应用带来 Web 特有的文化，以及用户行为体验给开发方带来更大压力。

（2）无法训练冲浪者，因此，Web 应用需要具有更好的自我解释能力。

（3）网上购物应用越来越多，这些 Web 应用的目的是销售产品，而用户并无法像商店那样直接看到商品。

（4）在 Internet 环境中，有大量的匿名用户，因此，无法像与真正销售人员交流一样进行交流。

Web 应用的可用性，具体来说，主要体现在以下几个方面。

（1）Web 应用的设计能够使用户把知觉和思维集中在自己的任务上，可以按照自己的行动过程进行操作，不必分心在寻找人机界面的菜单或理解 Web 应用结构、人机界面的结构和图标含义上，不必分析考虑如何把自己的任务转换成计算机的输入方式和输入过程。

(2) 用户不必记忆面向计算机软硬件的知识。

(3) 用户不必为键盘鼠标的操作分心,操作动作简单重复。

(4) 在非正常环境和情景时,用户仍然能够正常进行操作。

(5) 用户理解和操作出错较少。

(6) 用户学习操作的时间较短。

现在 Web 应用普遍存在不同程度的可用性问题,对发达国家站点可用性问题的有关研究表明了以下几点。

(1) 90%的企业站点可用性较差。

(2) 70%的企业对其站点设计不够满意。

(3) 用户在商业站点上找到所要信息的概率只有 42%。

(4) 由于用户不能从站点找到所需信息,造成约 50%潜在销售额的损失。

(5) 网上购物者最终放弃寻找欲购商品的概率为 62%。

(6) 51%的站点可用性问题是由于未运用最基本的可用性原则。

(7) 依据基本的可用性原则对 30 个 B2B 站点进行测评,无一通过。

由于 Web 用户一般都没有耐心,浏览也是非线性的,所以,他们不喜欢花哨的 Web 页面,不喜欢导航不清的 Web 页面。他们虽然说不清什么样的 Web 页面能够令他们满意,但是当一个 Web 应用采用统一的 Web 应用配色风格、统一的导航方式、统一而且明确的图示并且重点突出、内容新颖的时候,用户自然而然地就提升了满意度,Web 应用的可用性自然就好了。

11.2.1 Web 可用性原则

可用性是 Web 应用在特定使用环境中为特定目标用户所使用,从而快速、有效、满意地完成特定任务的程度。高可用性的用户界面使用户全神贯注于正在进行的工作,不用花费很多心思考虑如何使用该 Web 应用。

根据 Nielsen 的定义,Web 应用可用性属性如表 11.1 所示。

表 11.1 Web 应用可用性属性

可用性属性	描述指标
易学性和易记性	新用户能多快学会有效地完成一些基本工作;老用户能否清晰地记得如何再次有效使用,还是必须重新学习所有的东西
有效性	能否帮助用户准确地找到所要的信息,或者是用户通过 Web 应用完成特定任务的准确程度;能否让用户完整地找到自己所要的信息或实现特定目的
效率	用户使用该 Web 应用能多快地完成任务
容错度	在使用 Web 应用的过程中,用户出错的频率是多少,这些错误有多严重,用户是如何从错误中恢复的
用户满意度	用户喜欢使用该 Web 应用的程度

对 Web 设计的可用性的原则还没有清晰的界定,但很多可用性专家、用户体验设计师、交互设计师(如 Cooper、Robert Davis、Paul Laroche 等)提出了许多值得借鉴的看法,其中最著名的是 Steve Krug 的 Web 可用性三大定律。

(1) 不要让我思考(Don't make me think)。

(2) 点击多少次都没关系,只要每次点击都是无须思考、明确无误的选择。

(3) 去掉每个页面上一半的文字,然后把剩下的再去掉一半。

一个设计良好的Web应用,首先,应该让用户第一次访问Web应用时不需要思考就能对该Web应用有充分的认知和了解,并能做出初步的思维判断。如果用户在第一次访问该Web应用的时候弄不清楚自己的位置,不知道Web页面最重要的是什么、他/她该从哪里开始访问的话,那么很难保证用户会再次光临该Web应用;其次,对于Web页面的设计,需要建立合理的页面布局,从而降低用户的理解难度,即在每个Web页面上建立清楚的视觉层次,尽量利用习惯的用法,把页面划分为明确定义的区域,明显标识可以点击的地方,最大限度降低干扰,等等;再次,Web页面上应该要省略掉不必要的文字,只有去掉了没用的东西,才能让有用的信息更加突出,才能让页面更加简洁,才能让用户在每个页面上一眼就能看见更多的内容;最后,为Web应用设置清晰的导航图,如果用户在Web应用上找不到方向,用户是不会使用该Web应用的。

一般情况下,一个可用性良好的Web应用应该使得用户的操作尽可能以最直接、形象、易于理解的方式呈现在用户面前;使用符合用户习惯性行为;尽可能地为用户提供向导性质的操作流程;随时响应用户的操作。只有Web应用为用户提供了巨大的方便,用户才可能接受使用该Web应用。为了达到Web应用的高可用性,应当遵循如下几个方面的原则。

1) 别让用户思考

根据Krug的可用性第一原则,网页应当清晰且不言自明。当你创建一个网站的时候,你的工作就是避免问题 — 那些需要用户反复慎重考虑前因后果才能做出决定的选择。

如果网站的导航和结构不直观,产生的问题就会数量大增,且使得用户很难理解系统是如何工作的,怎样才能从A点跳转到B点。一个清晰的架构,中等强度温和的视觉引导,易于识别的链接,可以帮助用户找到实现目标的途径。

让我们来看一个案例,以前的Beyondis.co.uk(见图11.1,现在已经没有了)宣称自己是"超越栏目,超越产品,超越分布"的。这是何含义呢?自从发现了用户倾向于"F"模式的网页浏览习惯,以上提到"栏目、产品、分布"是用户浏览网页时,首先必见的三元素。



图11.1 Beyondis.co.uk

虽然设计本身非常简单且直观，但用户仍然需要去找寻才能明白这个网页是做什么的。这就是所谓的不必要的问题。设计帅职责是要让问题降到0。具有视觉效果的解释已经放在右边。只要交换左右模块的位置，可用性就会增加。

ExpressionEngine.com（见图11.2）使用了与Beyondis.co.uk非常相似的结构，但避免了不必要的问题。更进一步的是，右侧的宣传性的口号起到了效果，用户们会选择去了解或者购买该产品。

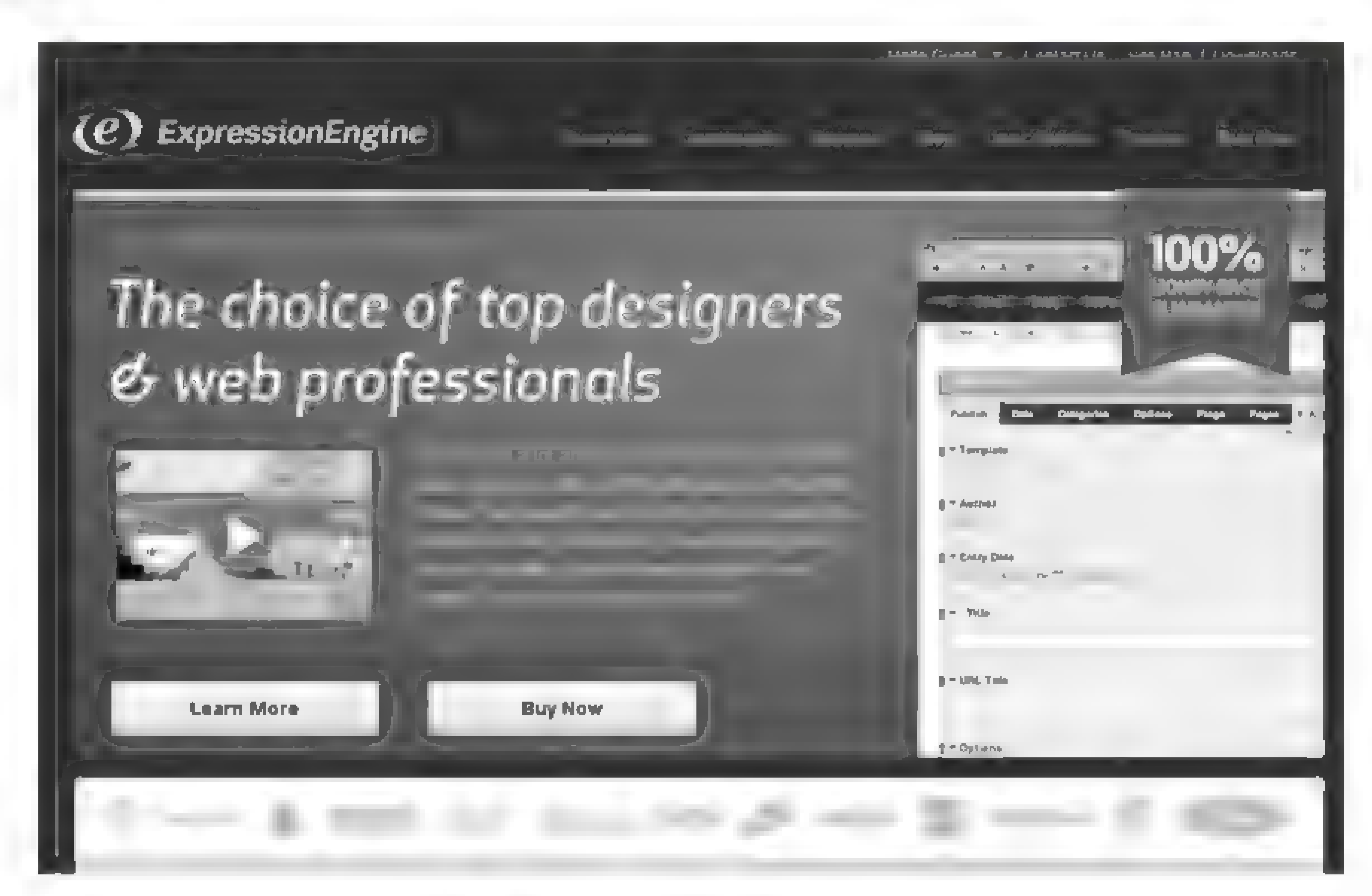


图11.2　ExpressionEngine.com

通过减少认知负荷，可以使访客更容易获取系统背后的思想。只要做到了这一点，就可以理解为何这个系统是有用的，而用户又是怎样从中获益的。如果人们在你的网页上迷路的话，他们是不会使用你的网站的。

2）别浪费用户的耐心

在任何一个想为用户提供服务或者使用工具的项目中，尽量使门槛降低，对用户的要求减少。一项服务要求用户付出的越少，越有可能被一个随机进入的访问者真正尝试。如果不用填那些他们以后都不会再次用到的长长的网页表格，首次来访的用户都会愿意尝试服务。请让用户自由浏览网页，让他们不用交换私人信息就能尝试你的服务。强迫用户填写电子邮箱地址来测试用户特征是不合理的。而用户们如果在看到产品之后被要求留下电子邮箱地址的话，他们可能是愿意的。

大众点评网的快速注册页面是一个用户友好的极佳例子，如图11.3所示。它清晰易懂，且几乎不向访客索取任何东西就能实现用户的注册，而且它还可以用其他网站比如开心网的账号直接登录。这也是在你的网站应当提供给用户的体验。

如果要理想化地去除所有的障碍，首先就是不需要贡献些什么或者填写注册。仅仅一个用户注册表本身就足以阻碍用户在网站的随意浏览行为，且会对网站浏览产生很大的不利影响。



图 11.3 大众点评网快速注册页面

3）抓住用户的注意

因为网站都是通常既提供静态的内容又提供动态的内容的，一些用户界面就会比另一些更加吸引人。很明显，图像比文本更吸引眼球——就好像加粗的句子比未加粗的更容易引起注意。

人类的眼睛是高度非线性运动设备，网页用户能够直觉地识别边界、模式和运动，这是为什么视频广告特别容易引起反感的原因。但是从市场营销的角度来说，它们的确完美地吸引了用户的注意。

Humanized.com 很好地利用了焦点原理。如图 11.4 所示，这个页面上直接呈现给用户的视觉元素只有一个“free”，非常吸引注意力，非常简洁且信息传递单纯。细小的线索给用户提供了充分的信息去找到“free”的产品。

4）尽量使特征明显呈现

当代网页设计总是嘲笑用鲜明视觉效果的大按钮指示用户：第一步——第二步——第三步……但是从设计的角度来说，这些元素事实上并非化石。相反，这些导航是极其有效的，因为他们能够以一种非常简单和友好的方式带领网页的浏览者在网站内容间穿梭。

Dibusoft.com 将视觉的宜人性和清晰的网站结构相结合，如图 11.5 所示。这个网站将主要的导航选择都放置在一眼能够看见的位置上，尽管这些导航的颜色有些太浅。

让用户看清楚功能的合理性是用户界面成功的基准。这个是否达到了，实际上并不重要，重要的是内容是否被很好地理解，而用户是否觉得他们与这个系统的交互非常舒服。

5）有效书写

由于网站与打印出版不一样，它需要与用户喜欢的书写方式相匹配，且与浏览习惯相契

合。鼓吹浮夸的文字将不会被阅读，大段没有图像、标粗或者斜体关键字的文本将被忽略，夸张的语言将被忽略。

应该避免太过于搞笑或自作聪明的名字、市场导向的名字、公司名或者不被熟知的技术名词。例如，如果你描述了一种服务，需要用户注册一个账户，“注册”比“就从这儿开始吧！”要好，而“就从这儿开始吧！”又好过“探寻我们的服务”。

Eleven2.com 直击要害，没有华丽的语辞，没有夸张的陈述，取而代之的是各种价格，如图 11.6 所示，而这正是用户来此所需要的。

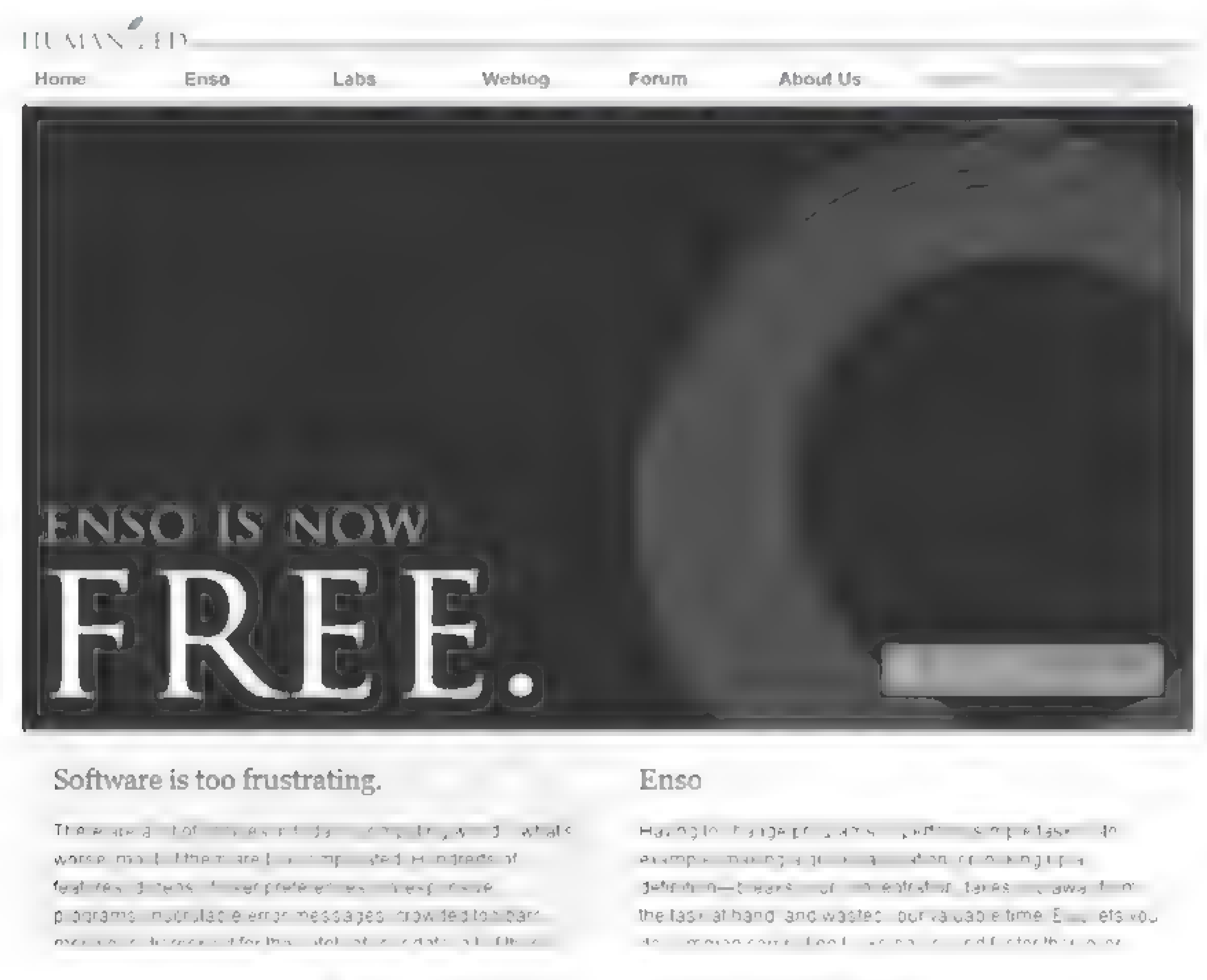


图 11.4 Humanized.com 利用了焦点原理

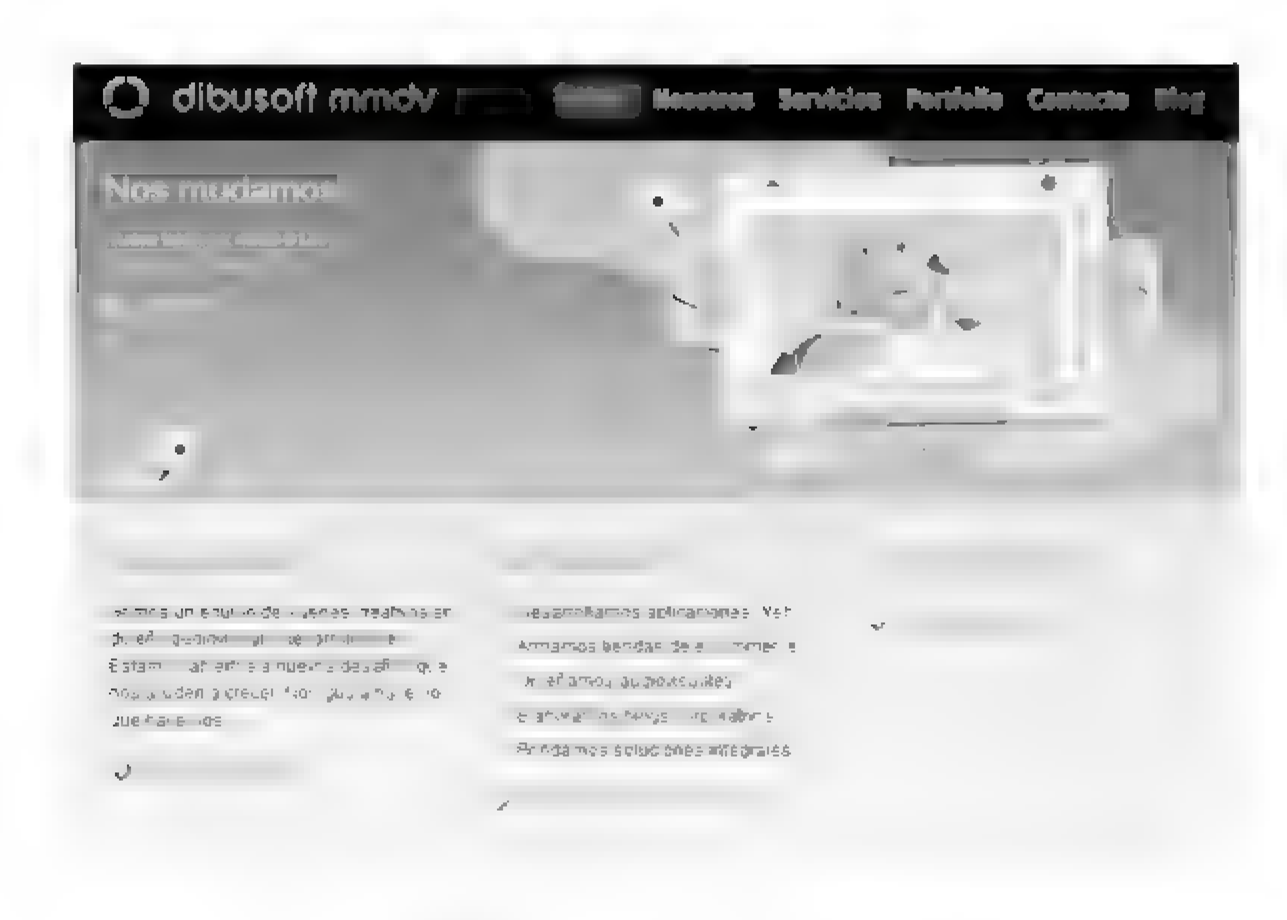


图 11.5 Dibusoft.com

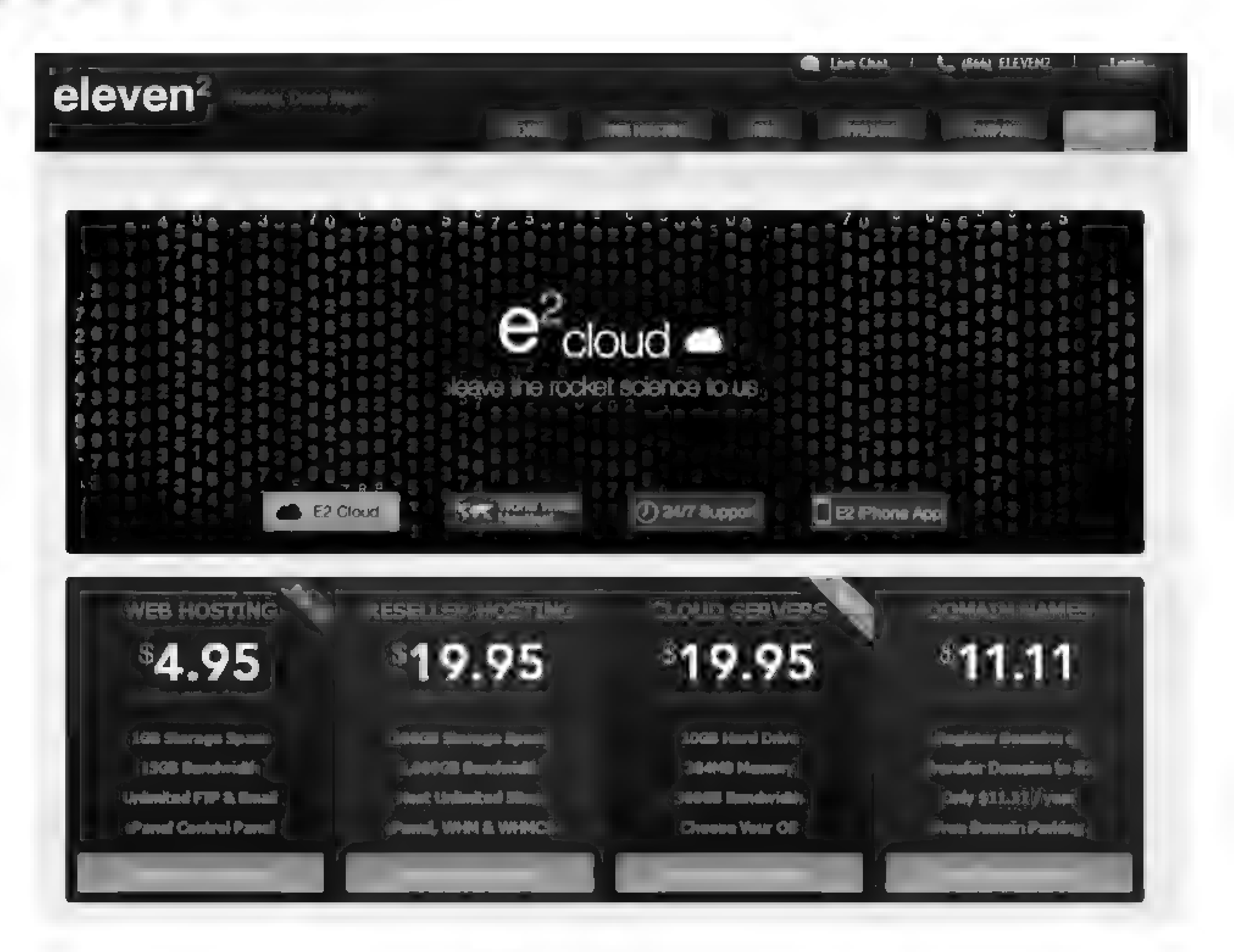


图 11.6 Eleven2.com

有效书写的优化解决方案包括：①使用简短的语句（直击要害，越快越好）；②使用铺陈的方式（将内容分类，使用多层标题，用视觉线索和树状图）；③使用平白直观的语言（一个宣传口号不用听上去像广告；给用户一些理性和客观的理由，让他们驻足在你的网站，享用你的服务）。

6）尽量简洁

“简洁”是网站设计的首要原则。很少有用户们驻足一个网站是因为喜欢它的设计，通常情况下他们是在找寻他们需要的信息，当然设计为他们提供了寻找帮助。尽量简洁，而不是复杂。

“人人网”为网页访客提供了一个整洁简单的设计。如图 11.7 所示，从页面上你可以清晰地识别出导航、标题、内容区域和脚注。注意，页面上的图标都可以清晰地传递信息。只要将鼠标悬浮在图标上，更多的信息就自动呈现出来。

从用户的角度出发，一个好的网页应当是纯文本的、没有广告的，内容与用户寻找目标密切相关。这也是一个方便打印的网页带来用户良好体验的原因之一。

7）别怕留白

事实上，在网页上留出空白区域的好处怎么估计都不过分。它不仅使网页访客的认知负荷减少，而且更容易获取网页所呈现的信息。新用户浏览网页时要做的第一件事情，通常是扫视全页，将内容区域在心理上划分成合适的组块，然后再对信息进行加工。

复杂的结构不易于阅读、扫视、分析和使用。可以通过两种方式进行优化：①用明显的竖线将两个区域隔开；②使用一些空白达到这种效果。通常比较好的选择是②，即留出空

白。分层减少页面复杂感(Simon 法则)：视觉上的层次感觉越好，页上的内容信息就越容易被获取。

空白区域能起到很好的效果。Cameron.io 用空白区域作为设计的主打元素，如图 11.8 所示。这样就使得主要信息被层次鲜明地突出了。



图 11.7　人人网

8）用“可视化”语言有效交流

Aaron Marcus 在“有效的视觉表达”一文中，提到的三个基本原则之一是“视觉语言”，即用户在屏幕上所看到的内容。

组织：为用户提供清晰稳定的概念结构。一致性、页面布局、模块关系和页面导航是组织中的重要概念。同样的表达方式和规则适用于所有元素。经济：尽量少使用视觉元素。需要权衡简约、清晰、区别性和重点突出 4 个要点。简约是指只有表达需要的重要元素才可以被呈现。清晰是指所有的组件都应当与它们传递的意义相吻合，表达要清晰，不要引起歧义。区别性非常重要，它要求所有元素都应当是独一无二的。重点突出是指关键的元素要能够被认知轻易捕获。

表达：使用用户能力能够接受的表达方式。为了使得表达顺畅，用户界面必须在可读性、易读性、结构、象征性、许多的观点、颜色和材质效果之间找到一种平衡。在一个页面里最多只能用 3 种字体，最多只能用 3 种字号，文本每行最多呈现 18 个字或者 50～80 个字符。

9）规范是我们的朋友

使用传统的元素设计出的网站并非索然无趣。事实上，传统规范非常有用，因为它们减少了学习的周期且节省了去收集有效性的精力。例如，如果所有网站对于 RSS 源都启用不

同的视觉特征，那么这将是可用性的一个梦魇。这就如同数据的规范整理，或者对于商场货架的规律摆放一样。

如果遵从规范，那么你将获得用户的信心、信赖和信任，且证明你是可靠的。遵从用户的希望，理解他们对网站导航、文字结构、搜索栏位置的期望等。（参考 Nielsen：Usability Alertbox）

在可用性测试方面一个典型的例子是：将网页翻译成日语（假设你的网站用户不懂日语，例如，使用 Babelfish），如图 11.9 所示，然后请可用性测试人员在不同语言的网页中寻找一些内容。如果很好地遵从规范，那么，尽管用户对于这种语言一窍不通，还是可以找到一些不是特别特殊的目标内容。Steve Krug 建议仅仅在你确认自己有了更好的想法时再去创新，但是如果没有的话，好好遵守现有规范。



图 11.8　Cameron.io 采用空白区域

10）早测常测

可用性测试通常是开发团队在设计和启动 Web 应用运行的最后才进行的事情。其实，可用性测试是启动一个 Web 应用的非常关键的一部分。早测常测（Test Early，Test Often，TETO）原则是指可用性测试应该从设计阶段就开始，然后不断进行测试，直到 Web 应用确实启动运行位置。TETO 原则可应用于任何网页设计，原因是对于现有布局的重要的问题和细节，可用性测试总能提供关键的信息。

测试不要做太迟、太少，或是为了不合适的理由而做。"不为不合适的理由测试"的意思是，许多设计方面的决策是为着当下的，不能笼统地宣布某些布局方式就一定优于其他，原因是你需要从一些特殊的角度来进行权衡，如考虑需求、投资者的利益、预算等。

根据 Steve Krug 的研究，测试一个用户要比一个都不测好一倍，且在项目启动之初测试一个用户要比项目告罄的时候测试 50 个要好得多。根据 Boehm 的法则，在需求和设计活动中错误非常常见，而发现得越迟，代价就越昂贵。

测试是一个迭代的过程。这意味着你需要设计些东西，然后接着就测试、修正，然后再测试、再修正，迭代进行。也许第一次迭代有些问题不能被发现，因为这些问题可能被其他问题所覆盖，用户们在其他问题上就已经被绊住了。



图 11.9 Babelfish 实例：日语版 amazon.com

11.2.2 可访问性

据统计，世界上大约有 20% 的人群至少有一种或多种身体残疾或身体缺陷，在这些人中有越来越多的人使用 Web 作为一种沟通与信息媒介来进行交流、信息获取、工作、学习等活动，这就要求 Web 应用对他们来说也是可访问的、可用的。究竟什么是可访问性(Accessibility)，很难给出严格的定义。而大家通常所理解的可访问性指的是 Web 应用中的内容对所有用户(包括残障用户)的可阅读和可理解性。可用性和可访问性针对的是 Web 应用的两个不同方面，因此很难给出两者之间的关系。而由于良好的可访问性对所有用户来说都是好事，所以经常把可访问性归为可用性的一类。

视力障碍是与 Web 应用可用性联系最为紧密的问题。视力障碍会直接影响到用户对 Web 应用的观看与浏览，极大地阻碍用户通过 Web 应用获取信息，因此，对于视力障碍的用户访问 Web 应用，一般可以通过使用屏幕阅读机等输出设备来帮助他们阅读 Web 页面内容。行动、听力或认知障碍也在 Web 可用性中扮演着重要的角色。对于老年人而言，在一个基于鼠标的导航系统中，一些点击元素太小也可能导致问题，例如可点击的图片或菜单太小。目前，Web 应用一般都忽略了听力方面的问题，随着多媒体技术的日益发展，其重要性也日益突出。特别是认知能力，例如逻辑思维能力、复杂上下文的理解等这些内容与 Web 应用开发人员密切相关，却不能期望用户具有如此的能力，因而制作简单的导航功能可以帮助所有的用户，特别是对于这类有认知障碍的用户群而言非常有用。

2008 年，国际组织万维网联盟发布了 WCAG(Web Content Accessibility Guideline)2.0。WCAG 2.0 定义了一系列指导方针，以提高 Web 内容对于残疾人的可访问性。在 WCAG 2.0

之前，解决可访问性问题的方案是各不相同的，一些常见问题都没有标准的解决方案。而借助 WCAG 2.0，就可以使用一些常见的特性来确保应用程序可以访问。

11.2.3 可用性和可访问性模式

按照亚历山大的思想，一个独立的模式应该包括三个部分：问题（或者目标）、解决方案以及上下文环境。一种模式应以直观图的形式提供一个典型的例子，帮助理解使用该模式。

可用性模式和交互设计在研究和实践中进行了广泛的讨论，并且已经出现了一些框架。典型的如 Borchers 开发和详细说明了一个模式框架，该框架主要用于互动音乐展示。Folmer 等发布了一个有关软件架构模式的框架，Graham 发布了网络可用性模式语言。在这些框架中，有时会提及可访问性，但是没有进行完整的论述。其中，值得关注的是，Helmut Vieritz 等人在可用性模式和可访问性模式之间建立了关联映射关系，这样就可以将两类模式共同进行考虑。

按照 Helmut Vieritz 等人的思想，有意义的序列（Meaningful Sequence）、键盘（Keyboard）、焦点顺序（Focus Order）、标签或提示（Labels or Instructions）、错误标识（Error Identification）等可访问性模式都可以关联到可用性的 Wizard 模式。

以“幸福密码”网站为例，其注册页面如图 11.10 所示。这是一个典型的向导式模式，用户按照提示一步步进行操作，当前所在步骤、已经完成的步骤以及还需进行的步骤都会显示出来，而用户在向导的帮助下进行一系列的选择。一般来说，用户向导都是通过一些表单来收集用户信息，用户通过点击一些按钮来完成其中的单个步骤。为了实现可访问性，表单中的输入框必须要有明确的标识，而且能通过键盘操作逐个激活，当出现错误时，也必须有明



图 11.10 “幸福密码”网站注册页面

确的错误提示。在这里面所涉及的可访问性模式主要包括有意义的序列、键盘、焦点顺序、标签或提示、错误标识，如图 11.11 所示。



图 11.11　页面中涉及的可访问性模式

11.2.4　移动可用性

移动可用性是指用户通过使用移动设备访问 Web 应用，并达到目的。随着移动互联网的日益完善，移动设备的日益廉价，越来越多的移动设备都可以快速地访问 Web 应用。

1. 移动可用性面临的问题

必须要关注到移动可用性存在着的诸多问题，主要表现在如下几个方面。

(1) 屏幕小。为了便于移动和携带，就必然需要小尺寸。小尺寸屏幕意味着在相同时间段内可供视觉的选择范围更小，更多地要求用户依靠他们的短时记忆来建立对在线信息空间的理解。

(2) 输入不便。由于设备体积小，可供输入的键盘面积有限，难以多个手指同时操作，导致文字输入很慢且容易出错。

(3) 系统资源有限。由于移动设备的体积与网络带宽的原因，使得其硬件和网络等系统资源十分有限。

(4) Web 应用的不良设计。因为 Web 应用大多是为桌面系统设计的，所以在可用性方面，并没有遵从移动设备应用的设计指南。

(5) 成本比较高。移动网络的收费远高于那些固定线路的互联网接入服务。

2. 移动可用性设计原则

现阶段虽然拥有了很先进的技术，但是距离良好的用户体验还很遥远，而且Web应用只是可以通过移动设备访问还远远不够，即使能提供全功能浏览器的高性能移动设备，其可用性仍然很难与个人计算机媲美。要想提高移动可用性，除了要做到移动设备的智能化、移动速度的高速化、资费的低廉化外，还必须在设计上下工夫。以下是一些移动可用性的设计指导原则。

① 使用用户熟悉的与基于个人计算机的Web应用一致的名称作为标签。

② 每页都包含到首页的链接和返回链接，并且最好在页面的顶部和底部均放置此类链接。

③ 只显示相关信息，同时，信息组织要尽量精练简洁，不要将重复的内容或链接放置在页面上。

④ 提供图片预览和铃声试听功能。

⑤ 提供合理的导航：提供一致的导航方法，导航层次不要太深，尽量使用与用户所熟悉的移动设备菜单相类似的简单层次。

⑥ 提供清晰的分类。

⑦ 提供站内"搜索"功能，且其位置要合理，易于用户快速找到。

⑧ 考虑到移动设备自身屏幕小、无线网速局限性等方面的限制，Web应用上放置的图片要适量。

⑨ 移动设备的输入机制不太方便，太多的输入会严重影响用户的使用效率，从而影响用户的体验，所以尽量提供选择，以减少用户输入。

目前的移动应用多个平台共存，主流的操作系统有Android、iPhone OS、Windows Mobile和Symbian等。

11.2.5 Web可用性工程

完成可用性需要一系列的动作，需要和所采用的Web应用开发过程一致并可集成，就出现了可用性工程。自20世纪70年代可用性工程产生以来，大量的实践已经形成了严格地、科学地分析和评估产品可用性的原则、生命周期和方法。可用性工程的三大原则是用户与任务、实证测量和不断设计以趋完美。可用性工程经过几十年的不断发展，形成了大量的可用性分析测量方法，比如启发式评估法、边想边说法(Think Aloud)、中心小组法(Focus Group)、测试法、模拟法等。这些方法既有定性测量方法，更有定量测量工具，灵活运用这些方法，可以从各个方面全面地测量产品的可用性。

1. 可用性设计考虑因素

Web可用性工程是将可用性工程技术运用到Web应用设计中，是Web应用设计人员以用户为中心创建Web应用而不再只是重视技术，但是由于Web应用的特殊使用条件，Web可用性设计时需要考虑以下特殊因素。

(1) 用户定义及使用环境。访问Web应用的用户可能来自世界各地，这些用户之间会有不同的语言和文化背景，他们所使用的技术平台也可能存在很大的差别。Web开发人员

很难控制甚至很难知道Web应用的实际使用者到底会是谁。所以，在Web应用设计中应当充分考虑用户背景及使用环境的复杂性和多样性。

（2）市场和竞争者分析。Web应用的竞争者可以直接通过网上查询发现其对手的情况。由于竞争对手的Web设计和实现手段都相当透明，所以对于Web应用进行市场和竞争者分析时，往往可以迅速得到大量有益的和实际可用的信息。

（3）需求、任务分析和目标定义。Web应用的工作方式应当与用户实现任务的习惯相符合。例如，主要的用户任务应当用明显的方式予以表现，页面和页面之间的流程关系应当符合用户完成任务的顺序。

（4）Web实施技术手段。Web应用在开发完成后瞬间就可以发表而被千万用户使用。因此，Web设计时要选择适当的实现技术，这不仅考虑到服务器能力、页面生成方式，而且也要考虑到用户浏览器、网络速度等因素，以保证Web应用的可用性。

2. 可用性评估

Web应用的用户一般不是Web应用专业人员，所以Web应用设计人员不可以假设用户和自己具有一样知识结构、习惯、思维，并且也不可以假设遵循设计指南就足以确保良好的可用性，所以需要对Web应用进行评估，检查用户是否能够使用以及是否喜欢使用这个产品。可用性评估指的是系统化的数据收集过程，目的是了解用户或用户组在特定环境中，使用产品执行特定任务的情况，从而发现产品中的可用性问题。

因此，对于Web应用可用性的评估必须综合各方面因素，避免仅从一个因素进行评估而带来的片面性评估结果。而用户在浏览Web应用时的各种各样的用户行为，恰恰从不同角度反映了用户所浏览Web应用的可用性。因此，以用户为中心，将评估一个Web应用的可用性转向研究浏览这个Web应用的用户行为上来，将这些用户的浏览行为以自动记录的方式记录下来，在必要的时候提供给可用性专家使用。这就是基于客户端用户行为记录的Web应用可用性的评估方法。

可用性评估方法一般可以分为5类：用户模型法、用户调查法、用户测试法、启发式评估法和认知性遍历。

1）用户模型法

用户模型法是用数学模型来模拟人机交互的过程。它把人机交互的过程看做是解决问题的过程，认为人使用Web应用是有目标的，而一个大的目标可以被细分为许多小的目标。为了完成每个小的目标，又有不同的动作和方法可供选择，每一个细小的过程都可以计算完成的时间。这个模型可以预测用户完成任务的时间，这种方法特别适合于无法进行用户测试的情形。在人机交互领域中最著名的预测模型是GOMS模型(1983年由Card，Morgan和Newell提出在交互系统中用来分析建立用户行为的模型。它采用“分而治之”的思想，将一个任务进行多层次的细化)。

2）用户调查法

用户调查法包括问卷调查法和用户采访法。问卷调查是一种结构化的调查，其调查问题的表达形式、提问的顺序、答案的方式与方法都是固定的，而且是一种文字交流方式，因此，任何个人，无论是研究者还是调查员都不可能把主观偏见代入调查研究之中。调查的统计结果一般都能被量化出来。由于问卷调查结果便于统计处理与分析，现在有

大量的相关统计分析软件可以帮助进行数据分析，有些甚至能直接帮助设计问卷的内容，方便实施和分析，也方便进行数据挖掘。用户采访法是一种非结构的调查方法，预设提纲进行口头采访，用户也可以向采访人发问，提出一些想法，所提的问题和提问的顺序受用户回答的影响。采访时可以先从大范围的问题开始，再逐步了解其中的细节与延伸性的问题。

3）用户测试法

用户测试法是最有效的可用性评估法。用户测试时测试人员邀请用户使用设计原型或产品完成操作任务，并通过观察、记录和分析用户行为和相关数据，对界面可用性进行评估。因此，用户测试法最能反映用户的需要，能了解不同问题的重要程度，适合于产品界面和界面设计中后期界面原型的评估。用户测试可分为实验室测试和现场测试，实验室测试时在可用性测试实验室里进行，而现场测试时，由可用性测试人员到用户的实际使用现场进行观察和测试。这种方法相对来说需要投入较多的时间和人力，能找出主要的问题但不易发现全面性的细节问题。

4）启发式评估法

启发式评估法（Heuristic Evaluation）也称为专家评审法，是由有经验的可用性专家来评估 Web 应用的可用性。启发式评估法使用一套相对简单、通用、有启发性的可用性原则来进行可用性评估，通过观察界面来指出其中哪些设计好，哪些设计不好，其目标是要发现界面设计中的可用性问题。

启发式评估是非正式的可用性检查技术，它包括 3～5 个用户或评估者，一些评估者在设计中要检查与之匹配的可用性问题的可用性原则，每个评估者对界面独立进行评估，对发现的可用性问题的严重程度要进行评分，最后给出设计修改建议。其优点是以最少的时间与资源找出最多的问题，并能在 Web 应用开发初期立即修正设计上的问题，分析的范围广。缺点是因为主要由可用性专家进行分析，其想法不一定等同于实际使用者，所以其客观性不足，而且不一定能对问题的严重程度分级，无法得知实际用户的期待与可能的潜在需求。

5）认知性遍历

认知性遍历法试图想象出人们在第一次使用某个 Web 应用时的想法以及所采取的动作，它的大致流程如下：已经有一个原型或对于页面的详细描述或一个真正的 Web 应用产品，同时，知道可能的用户是谁。此时，可以选择产品所能支持的某个（些）功能来进行评估。评估的具体过程是把用户在完成这个功能时所做的所有动作描述成一个令人可以信服的故事。为了使得这个故事可信，针对于用户所做的每一个动作，必须要能够证明：根据用户的知识水平以及页面上的各种信息提示及反馈，用户做出该动作是合情合理的。认知性遍历的核心是对假定的用户所采取的每一个动作进行质疑，看它的发生是否合乎情理。认知性遍历只适合于评估一个产品的易学习性，因为它考虑的是用户在第一次使用 Web 页面时的想法和行为，但不太容易发现使用效率方面的可用性问题。

从适用阶段、信息和数据、成本、耗时、测试人员以及综合评价几个方面对以上几种可用性评估方法的相互对比，如表 11.2 所示。

表 11.2 几种可用性评估方法对比

评估方法	适用阶段	信息、数据	成本	耗时	测试人员	评价
用户模型法	始终	客观、定量数据和定性扫描	低	省时	设计人员	适用于无法进行用户测试的情况，侧重于预测用户的执行性能
用户调查法	完成以后	主观、定性	低	省时	用户	测试成本低，当前比较通用的可用性评估方法
用户测试法	始终	客观、定性	高	费时	专门可用性测试人员和用户	全面、准确，近似于真实的使用环境，测试时间和资金成本较高，易受主观影响
启发式评估法	设计阶段或者完成阶段	主观、表达描述	低	中等	专家测评者	对于测试者要求比较高，受主观因素影响较重，缺乏固定统一的标准
认知性遍历	设计阶段	主观、表达描述	低	中等	专家评审者和典型用户	测试的准备要求比较高，须具备原型或者详细设计说明，对于测试人员的要求高

3. 可用性工程活动

目前，很多大公司在开发Web应用时包含某些可用性工程过程。这类开发过程中，可用性工程涉及的人员主要为4类：系统分析师、页面设计人员、可用性专家和实现人员。可用性工程过程一般包括如下活动。(更详细的描述参考Jakob Nielsen的《Usability Engineering》)

1) 需求分析

了解用户，以提供用户的基本模型，进一步控制可用性工程的后续阶段的开发。通过竞争分析，了解竞争的Web应用的优劣，提供给要构建的Web应用一些重要实践。成功可用性工程的分析阶段的一个重要目的是进行分析员或可用性专家定性和定量地确定可用性目标。例如，一个信息型Web应用的目标是提供内容，其定性目标可以如“要成为当地最知名的旅游指南”，而定量的目标则如“70%的测试人员搜索商品X和Y的不成功率<3%”。以使用为中心(Usage-centered)的任务分析，确定任务的优先级，明确可用性应该更多地强调易学性还是易用性，例如，从网上订购冬季旅游套餐很可能需要易学性，而经常进行标准交易的网上银行则需要易用性。在分析任务的同时，还需要了解使用Web应用的设备，如PC、PDA还是智能手机等，了解用户使用Web应用的条件，如移动、静止、休闲还是紧张。在这种情况下，通常采用场景技术以确保所有重要的可用性因素都被考虑。用户直接或间接参与设计。

从可用性工程的角度而言，需求分析可以概括为：了解用户兴趣模型；进行任务分析，设计场景和用例模型；各种平台的规格说明，如设备、浏览器和插件等；定性和定量分析可用性目标。

2) 设计

页面设计人员根据需求分析结果，开发一个整体用户页面的概念模型，定义Web应

用的基本结构，辅助开发者了解整个应用。这个模型主要关注基本表示、使用故事和导航与交互原则。可以采用用户参与法以满足目标用户群的需求，然后可用性专家对模型进行评估。

设计过程的结果为：概念模型、详细设计、测试结果与设计决策。

3）实现

实现阶段除了实现人员外，可用性专家也起着重要的作用。可用性专家需要不断检查Web应用的一致性，进行持续观察，以及如果需求有变时指定开发策略。另外，用户参与能够保持随时给出Web应用是否满足他们期望的反馈意见。研究要安装的环境，以使实现出的Web应用能够适应要安装的环境。

4）运行

可用性工程的运行期目标是工具实际和长期的使用，确定用户使用系统用户体验，以便在新版本中能够体现。

4. 可用性工程的意义

可用性工程在Web应用设计中具有重要的意义，主要体现在以下几个方面。

(1) 首先，可用性工程原理明确了Web应用分析的原则，这是由Web应用分析和可用性工程的内在相似性决定的。由于Web应用相对于其他信息系统而言，更加强调其可用性，可用性就是Web应用的生命线，可用性分析几乎构成了Web应用分析的全部内容，所以在某种程度上，Web应用分析应该是可用性工程在Web应用产品中的具体应用。

(2) 可用性工程为Web应用分析指明了分析方向 — Web应用分析应该分析什么，这是Web应用分析应该首先解决的问题。可用性工程从可用性角度出发，告诉人们Web应用分析的核心应该为Web应用的有效性、高效性和满意度。

(3) 可用性工程的生命周期理论说明，Web应用分析不是一个简单过程，而是具有复杂阶段的生命周期，只有遵循严格的步骤和方法论，才可以保证分析的准确性和可靠性。

可用性工程现有的方法极大地丰富了Web应用分析的方法，不仅可用性工程的具体方法对建立新的Web应用分析方法起到启示作用，而且很多可用性工程的现有方法可以直接应用到Web应用分析环境中。

11.3 总结与展望

在当今的Internet环境中，Web 2.0应用正在变得越来越流行，富客户端、大页面、大量JavaScript编码被广泛应用，给Web应用的性能问题带来了新的挑战。另一方面，自W3C相继发布WCAG 1.0和WCAG 2.0以来，可用性慢慢成为了大多数开发人员心中对Web的一个评判标准，越来越多的Web应用开始提供更好的可用性。但由于实际上的关注和投入不够，Web可用性仍有很长的路要走。

本章主要讲述了Web应用的性能和可用性问题，分析了影响Web应用性能的指标以及Web应用性能提升策略，阐述了Web应用可用性原则、Web可用性工程方法以及趋势。

在提升 Web 应用性能方面，出现了一些卓有成效的最新工作，典型的如 Google 提出的 Diffable。Diffable 能在浏览器加载页面时，比较相关文件（如 HTML 和 JavaScript 等）在服务器端和客户端缓存区的版本差异，只下载存在的差量，从而有效地加快页面加载速度。

在 Web 可用性方面，AJAX、UGC（User Generated Content）等技术和 WCAG 2.0 等标准将会对 Web 可用性产生更多影响。可用性工程持续改进，可用性模式和新的方法也会向更多特定目标领域或特定目标发展。而多模式交互、眼球跟踪、对用户注意力的感知等则给 Web 应用及其可用性提出了更高的挑战。

第12章 Web应用的安全性

随着计算机应用的普及和网络的不断发展，网络和信息的安全问题日益突出，网络入侵、攻击和病毒以及不良的有害信息给 Web 应用的安全性带来了严峻的挑战。伴随着 Web 信息和服务的可用性的提升，以及基于 Web 的攻击和破坏的日益增多，Web 安全风险达到了前所未有的高度。据统计，目前 Internet 上超过 60%的攻击针对 Web 应用，Web 安全已经成为 Internet 上的焦点，它关系到 Internet 的进一步发展和普及，甚至关系到 Internet 的生存。因此，Web 应用的安全是 Web 应用构建、运行、维护与管理过程中最重要的任务之一。

Web 应用安全性是指一种能够识别和消除网络上对 Web 应用的硬件、软件、数据等造成破坏的不安全因素的能力，凡是涉及网络上的信息保密性、完整性、可用性、真实性和可控性的相关技术和理论都是 Web 应用安全的研究领域。

广义上，凡是涉及网络上信息的保密性、完整性、可用性、真实性、确认性和可控性的相关技术和理论都属于网络安全领域。从用户角度看，Web 应用安全指的是用户的信息在网络上传输时受到机密性、完整性和真实性的保护，避免其他人或对手利用窃听、冒充、篡改、抵赖等手段侵犯用户的利益和隐私。而且，这些不安全因素越来越多：病毒种类的多样性，黑客技术的公开和有组织化，以及网络的开放性使得网络受到攻击的威胁越来越大。Web 应用的安全性主要包括操作系统平台的安全性、Web 服务器的安全性、Web 应用的安全性、信息传输过程的安全性、管理上的安全性。

12.1 Web 应用安全性特性

目前很多业务都依赖于互联网，例如网上银行、网上购物等 Web 业务平台，很多恶意攻击 Web 服务器的行为，想方设法通过各种手段获取他人的个人账户信息谋取利益。由于架构不同，使得 Web 应用的安全性与传统的应用程序的安全性有所不同，使它面临着更多的安全威胁，需要更严格的安全防护策略。

1) 开放性

Web 应用具有很强的开放性，导致了它易受攻击，这主要体现在三个方面。

(1) 状态信息的开放性。HTTP 协议是无状态的，Web 应用的开发人员需要自己记录程序运行的状态，并在客户端和服务器端进行保存和传输，这些信息对终端用户公开，容易被恶意更改和伪造。

(2) 源代码的开放性。Web应用的客户端程序大量由JavaScript等脚本语言编写,其源代码对用户可见。此外,还有些用Java编写的Applet程序,也可以通过反编译得到源代码,恶意用户可以通过修改源代码来进行攻击。

(3) 执行顺序的开放性。Web应用的执行通过多次页面请求来实现,恶意用户很容易跳过某些程序执行步骤,而直接去执行后面的内容,从而造成安全问题。

2) 信息流通的灵活性

与传统信息不同,Web包含了文本、语音、图像、视频等多种资源,一个Web页面可以引用来自多个服务器的资源,一个服务器也可以被多个不同的客户端访问,这使得Web应用更容易受到来自Internet的攻击。

3) 服务器容易受到攻击

计算机只要连接到Internet,Web服务器上的操作系统、系统配置都无法做到对病毒、木马等恶意程序的百毒不侵,任何操作系统都或多或少会存在漏洞或是缺陷,黑客或不法分子可以利用这些缺陷进行攻击、盗号等非法活动。

4) 开发人员的局限性

许多Web应用开发人员不知道如何开发安全的应用程序或是没有考虑到Web应用的安全性,在Web应用开发的过程中很容易忽略掉Web应用的安全性问题,但是不安全的Web应用在安全缺陷被利用时可能会出现灾难性后果。此外,Web开发技术日新月异,各种语言、框架和技术层出不穷,开发人员很难跟上这些新技术的发展,在应用这些技术进行开发时并没有真正地掌握和理解,导致开发出的应用存在缺陷或安全漏洞。

5) 底层应用软件漏洞众多

虽然Web浏览器非常易于使用,Web服务器相对而言易于配置和管理,Web内容也易于开发,但是其底层的软件却非常复杂,很难有软件能够做到不存在任何漏洞或缺陷,它们隐藏潜在的安全漏洞。

6) 用户错误操作

Web服务的客户通常在安全方面未经过训练或重视不够,这些用户通常并不具备安全风险的概念,也没有采取相应措施以消除不安全因素的知识或工具。

12.2 Web应用安全威胁

Web应用的威胁主要来自计算机病毒、木马、蠕虫等通过计算机网络危害计算机安全的恶意代码以及间谍软件等。除此之外,自然灾害(地震、海啸等)、意外事故(比如Web服务器进水、着火等)、人为攻击(黑客非法访问等)和网络协议缺陷(TCP/IP协议的安全问题等)等都会导致计算机网络的不安全性。

12.2.1 安全威胁种类

对Web应用安全威胁形形色色,种类繁多。

1) 计算机病毒

计算机病毒是未经授权而执行的程序,感染、潜伏、可触发、破坏、不可预见是计算机病

毒的基本特性。感染使病毒得以传播，破坏性体现了病毒的杀伤力。感染和破坏行为总是使系统或多或少地出现异常，因而频繁的感染和破坏会使病毒暴露，而不破坏、不感染又会使病毒失去杀伤力。计算机病毒具有自我复制和传播的特点，经常通过电子邮件、BBS、Web页面和即时通信软件等传播。

2）蠕虫

蠕虫是由一个主程序和一个引导程序组成。主程序一旦在计算机中得到建立，就可以收集与当前机器联网的其他计算机的信息。它能通过读取公共配置文件并检测当前计算机的联网状态信息，尝试利用系统的缺陷在远程计算机上建立引导程序。这种引导程序类似于“钓鱼”的小程序，它把“蠕虫”带入了它所感染的每一台计算机中。蠕虫程序能够常驻于一台或多台计算机中，并有自动重新定位的能力。假如它能够检测到网络中的某台计算机没有被占用，它就把自身的一个复制件（即一个程序段）发送到那台计算机。每个程序段都能把自身的复制件重新定位于另一台计算机上，并且能够识别出它自己所占用的那台计算机。蠕虫的行为具有主动攻击、行踪隐蔽、利用系统或网络应用服务漏洞的特性，危害巨大，通常会引起网络拥塞，降低系统性能，产生安全隐患，而且反复性大，破坏性强。近些年来，越来越多的蠕虫把木马的特性、病毒特性结合起来，功能更加强大，抗查杀能力更强，杀伤力更大，并且加入了DoS（拒绝服务）攻击，加剧了对网络的危害。

3）木马

木马，全称是“特洛伊木马（Trojan Horse）”，在Internet上，也叫黑客程序或后门，是不进行复制的带有恶意的程序。木马具有隐蔽性、顽固性、潜伏能力、危害性等特征。一般木马都有客户端和服务器端两个执行程序，其中客户端主要用于攻击者在远程控制植入了木马的目标机器，服务器端程序也就是木马程序，负责打开攻击的道路，两者需要一定的网络协议来进行通信。

4）恶意软件

恶意软件是指一类特殊的程序，它是介于计算机病毒与黑客软件之间的软件的统称。它通常在用户不知晓也未授权的情况下潜入系统，具有用户不知道（一般也不许可）的特性，激活后将影响系统或应用的正常功能，例如收集用户名、密码和敏感的银行数据等，甚至危害或破坏系统。其主要危害体现在非授权安装（也被称为“流氓软件”）、自动拨号、自动弹出各种广告界面、恶意共享和浏览器劫持等。当前，恶意软件的出现、发展和变化给计算机系统和网络信息系统带来了巨大的危害，尽管已经出现了很多防范措施，但恶意软件一般都很难甚至无法删除。恶意软件是影响Web应用安全性的重要因素，它一般都有强制安装、难以卸载、浏览器劫持、广告弹出、恶意收集用户信息、恶意卸载和恶意捆绑等特征。

5）安全漏洞

漏洞是在硬件、软件、协议的具体实现或系统安全策略上存在的缺陷，从而可以使攻击者能够在未授权的情况下访问或破坏系统。Web应用安全漏洞造成的危害在于它能被攻击者利用，继而破坏系统的安全特性。一般而言，安全漏洞对系统造成的危害主要有以下几个方面的内容。

（1）破坏完整性。攻击者利用漏洞入侵系统，对系统数据进行非法篡改。

（2）破坏可用性。攻击者利用漏洞破坏系统或网络的正常运行，导致信息或网络服务不可用。

(3) 破坏机密性。攻击者利用漏洞给非授权的个人和实体泄露受保护的信息。

12.2.2 安全漏洞

通常可能会出现如下一系列安全漏洞，这些漏洞的产生、预防与修复各异。

1) 不安全的配置管理

拥有安全软件和安全的配置是构建安全Web应用必要的条件。Web服务器负责提供内容，调用产生内容的应用程序：应用服务器为应用程序提供多种服务，包括数据存储、目录服务、邮件、消息等。而Web应用的服务器配置中往往存在很多安全问题，攻击者可以使用扫描工具检测到这些问题并加以利用，导致后端系统的攻陷，包括数据库和企业内部网络。

2) 欺骗攻击

欺骗攻击利用TCP IP协议的缺陷或用户的疏忽，让攻击者可以利用IP地址易于更改和伪造的缺陷，进行IP地址假冒，从而达到攻击的目的。网络欺骗的技术主要有HONEYPOT和分布式HONEYPOT、欺骗空间技术等。主要方式有IP欺骗、ARP欺骗、DNS欺骗、电子邮件欺骗、源路由欺骗(通过指定路由，以假冒身份与其他主机进行合法通信或发送假报文，使受攻击主机出现错误动作)、地址欺骗(包括伪造源地址和伪造中间站点)等。IP欺骗利用了RPC服务器仅仅对信源口地址进行安全校验的特性，在使被信任主机丧失正常通信能力后，猜测出目标主机TCP的序列号，可以伪装成被信任主机而建立起与目标主机基于地址校验的应用联结。

3) 不安全的加密

大多数Web应用将敏感信息存储到数据库或者文件系统中，通过使用加密技术来进行保护。常见的不安全的存储操作有：没有加密关键数据，密钥、证书和密码的不安全存储，秘密信息不恰当地在内存中存在，随机性来源不好，算法选择不好，没有包括对加密密钥交换的支持及其他必要的维护过程。可以通过将加密的使用最小化、只存储绝对必要的信息(如不存储信用卡号码，而让用户重新输入)、不存储加密后的密码而使用单项函数(如SHA-1来哈希密码)等措施保护自己不受此类攻击。

4) 错误的访问控制

访问控制是指Web应用如何给不同用户授予对内容和功能的不同的访问权限，是安全防范和保护的主要策略，主要任务是防止网络资源被非法使用、非法访问和不慎操作所造成的破坏。访问控制的检查在身份认证之后执行，用来决定被授权的用户能够实施的操作。一个Web应用的访问控制模型和Web应用提供的内容和功能紧密相连。用户会被分为具体的不同能力或权限的用户组或者角色。实现访问控制策略的代码应该是结构合理的、模块化的和集中化的，可通过代码检查和渗透性测试来验证其实现的正确性。根据整个应用程序的访问控制需求，建立文档描述的安全策略。定义何种类型的用户可以访问系统，每类用户可以访问哪些功能和内容。

5) 隐藏字段

在许多应用中，隐藏的HTML格式字段用来保存系统口令或商品价格。尽管其名称如此，但这些字段并不很隐蔽，任何在Web页面上执行"查看源代码"的人都能看见。许多Web应用允许恶意的用户修改HTML源文件中的这些字段。为了防止这种恶意篡改，应

该对返回 Web 页面进行验证。

6）未被验证的输入

Web 应用使用来自 HTTP 请求（有时来自文件）的输入来决定如何进行响应。攻击者可以篡改 HTTP 请求的任何部分，包括 URL、查询字符串、头部、Cookie、表单域和隐藏域，试图越过站点的安全机制。常见的输入篡改攻击有强力浏览、命令插入、跨站点脚本、缓冲区溢出、格式化字符串攻击、SQL 注入、毒化 Cookie 和混合隐藏域等。Web 应用的管理者和开发人员往往希望通过过滤恶意输入来保护自己。但是，Web 页面中存在多种信息编码方法且容易被破解。大量 Web 应用只使用客户端的机制来验证输入。客户端验证的性能和实用性很好，可以增强用户的体验并减少对服务器的无效流量，但是它没有任何的安全性，很容易被绕过。因此，必须使用服务器端的检查来对付参数篡改攻击。

7）后门和调试漏洞

开发人员常常建立一些后门并依靠调试来排除应用程序的故障。在开发过程中这样做可以有效地帮助开发人员解决 Web 应用中存在的缺陷，但这些安全漏洞经常被留在一些放在 Internet 上的最终应用中。一些常见的后门使用户不用口令就可以登录或者访问允许直接进行应用配置的特殊 URL，这样做经常会导致非法用户对应用程序的破坏行为。因此，需要在交付前进行严格的代码检查来避免遗留这种漏洞。

8）错误的认证和会话管理

认证和会话管理包括处理用户认证和管理活跃的会话。Web 应用通常使用用户 ID 和密码来进行用户认证，并使用会话来保存每个用户的请求流。因为 HTTP 协议本身并不提供这样的功能，所以 Web 应用环境须自己提供会话能力，妥善保护开发人员自己创建的会话 token 以免攻击者劫持活跃会话。合理使用自定义的或者通用的认证和会话管理机制，须注意一些关键问题，如通过限制密码长度和复杂度增加密码强度，限制并记录使用密码进行错误登录的次数及相关信息，完善的密码修改控制，以散列函数（Hash）或加密的方式存储密码，保护传输中的认证信息，使用 SSL 保护会话 ID，不允许以假名的形式列出账号，不用浏览器缓存认证和会话数据，建立可靠的信任关系，等等。

9）注入式攻击

注入式攻击包括对操作系统的调用，使用 shell 命令来调用外部程序以及通过 SQL 注入来调用后台数据库。用 Perl、Python 和其他语言写的脚本也可以被注入到 Web 应用中执行，许多 Web 应用使用操作系统和外部程序来实现它们的功能。当 Web 应用将 HTTP 请求中的信息作为外部信息的一部分进行传送时，它必须被仔细检查，否则，攻击者可以在信息和 Web 应用中注入特殊字符、恶意命令或是命令修饰符，而 Web 应用会盲目地把这些内容传送到外部系统去执行。

命令注入漏洞是用户在输入数据中插入操作系统命令导致软件在接受用户输入，并以此为参数调用某些操作系统命令或外部程序时，执行了意外的操作，例如程序调用了 finger 命令来进行查找，如果用户输入数据“name; ls-al”或者“name \r\n ls -al”，则命令 ls-al 也会执行。

SQL 注入是一种常见的注入形式，攻击者在一个 Web 表单的搜索字段中输入恶意 SQL 语句，欺骗服务器执行这些恶意 SQL，以查看或删除数据库中的内容，造成灾难性后果。SQL 注入漏洞危害越来越大，这主要是因为 Web 应用对数据库的依赖性日益增加。

避免注入式攻击的方式有：使用编程语言中的内部库来实现对外部程序的访问；仔细验证一些必须使用的外部调用的输入数据，保证不含任何恶意内容；将所有的数据当作参数，使用存储过程或预处理语句进行数据处理；保证 Web 应用只在它需要的特权下运行，如不以 root 运行 Web 服务器或以管理员账号访问数据库。

10）跨站点脚本(XSS)攻击

XSS 是指恶意攻击者在 Web 页面里插入恶意脚本，如在 JavaScript 脚本中加入恶意脚本，当用户浏览该 Web 页面时，这些脚本就执行，从而达到盗窃用户信息等目的。图 12.1 所示为其执行过程示意图，图中各序号代表的含义如下。

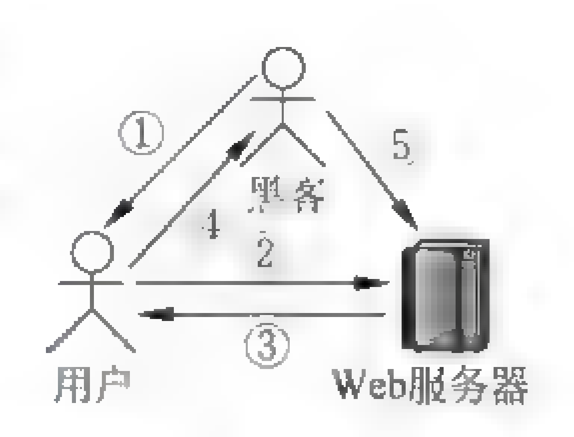


图 12.1 跨站脚本示意图

① 通过 E-mail 或 HTTP 将链接发送给用户。

② 用户将嵌入的脚本当作数据发送。

③ 浏览器执行脚本后，返回数据。

④ 在用户毫不知情的情况下，脚本将用户的 Cookie 和 Session 信息发送出去。

⑤ 攻击者使用偷来的 Session 信息伪装成该用户。

遵循严格的规范对所有的头部、Cookie、查询串、表项和隐藏项进行验证，是保护 Web 应用免受 XSS 攻击的好方法。也可以对用户提供的输出进行编码来防止被插入的脚本以可执行的形式传输到用户，例如，将一些特殊的字符转化成合适的 HTML 编码。

对于 Web 应用，坚决不要相信任何用户输入并过滤所有特殊字符，这样可以消灭绝大部分的 XSS 攻击。对于个人，保护自己的最好方法就是仅点击需要访问的那个 Web 应用上的链接。有时候 XSS 会在打开电子邮件、打开附件、阅读留言板、阅读论坛时自动进行，因此，当打开电子邮件或是在公共论坛上阅读陌生人的帖子时要多加留意。

11）更改 Cookie

Cookie 是一小段文本信息，伴随着用户请求和页面在 Web 服务器和浏览器之间传递，是一种保持 Web 应用"状态"的方法。

更改 Cookie 指的是修改存储在 Cookie 中的数据。Web 应用常常将一些包括用户 ID、口令、账号等的 Cookie 存储于用户的本地。通过改变这些值，恶意的用户就可以访问不属于他们的账户。攻击者也可以窃取用户的 Cookie 并访问用户的账户，而不必输入 ID 和口令或进行其他验证。因此，对于重要的信息，尽量采用 SSL 进行安全防护。

12）客户端嵌入程序

越来越多的 Web 应用在客户端通过插件扩展浏览器功能或添加小应用程序的方式，增强其客户端处理能力。但是，浏览器插件接口也成为某些病毒、木马、广告程序、间谍程序等恶意程序攻击网络的计算机的缺口；ActiveX 控件以可执行二进制代码运行于本地驱动器上，和其他本地应用程序具有相似的功能，也就意味着有害控件可以轻而易举地删除位于本地驱动器上的文件，或者将用户的私人信息发送到 Internet 上某台未知的服务器上；Java Applet 可以被放置在 Web 页面上，连同 Web 页面一起被下载到用户机器上，Java Applet 的安全模型的实现也不尽如人意。因此，使用这类程序时，要充分考虑到其可能引发的安全问题。

13）缓冲区溢出

缓冲区溢出是指当计算机向缓冲区内填充数据时，数据超过了缓冲区的容量，溢出的数

据覆盖在合法数据上。恶意用户攻击时，会向服务器发送大量数据以使系统瘫痪，该系统包括存储这些数据的预置缓冲区。如果所收到的数据量大于缓冲区，则部分数据就会溢出到堆栈中。如果这些数据是代码，系统随后就会执行溢出到堆栈上的任何代码。Web应用缓冲区溢出攻击的典型例子也涉及到HTML文件。如果HTML文件上的一个字段中的数据足够大，它就能创造一个缓冲器溢出条件。要避免溢出攻击，需要进行充分的安全测试。

14）物理路径泄露

物理路径泄露一般是由于Web服务器处理用户请求出错导致的，例如，通过提交一个超长或某个精心构造的特殊请求，或是请求一个Web服务器上不存在的文件。这些请求的一个共同特点是被请求的文件肯定属于CGI脚本，而不是静态HTML页面。另外，Web服务器的某些显示环境变量的程序错误地输出了Web服务器的物理路径，这种设计上的问题一定要避免。

15）目录遍历

通过对任意目录附加“.. ”，或者是在有特殊意义的目录附加“../”，或者是附加“../”的一些变形，如“..\”或“.. ”甚至其编码，都可能导致目录遍历。后面的几种情况比第一种情况常见得多。以前非常流行的IIS二次解码漏洞和Unicode解码漏洞都可以看做是变形后的编码。需要对Web服务器进行正确配置以避免这种攻击。

16）直接访问浏览

直接访问浏览指直接访问应该需要验证的Web页面。没有正确配置的Web应用可以让恶意用户以及非法用户直接访问包括有敏感信息的URL或者使提供收费Web页面的服务丧失收入。

17）不恰当的异常处理

当Web应用在正常操作的时候会频繁地产生错误情况，比如内存不够、空指针、系统调用失败、数据库不存在、网络不通等。正确地处理这些错误异常，给用户提供详细有意义的错误信息，给Web应用维护人员提供诊断信息但不能给攻击者提供任何信息。不恰当的异常处理会将详细的内部错误信息比如堆栈追溯、数据库清空及错误的代码等提供给攻击者，从而给Web应用带来很多安全问题。表12.1给出了不恰当异常处理漏洞的特征、可能产生的威胁以及解决漏洞的对策或方法。

表12.1 与异常处理相关的漏洞、威胁与对策

漏洞特征	无法使用结构化异常数据
	显示太多的信息给客户端
威胁或攻击	显示敏感系统或应用程序细节
	拒绝服务攻击
对策	使用结构化异常处理(通过try/catch)
	只有在操作添加值/信息时捕获和包装异常
	不要显示敏感系统或应用程序信息
	不要记录私人数据，例如密码

18）拒绝服务

绝大多数的Web服务器在正常情况下能够同时处理几百个用户的请求。一个攻击者可以从一台或多台客户机产生足够多的请求来耗尽应用程序提供的有限资源。这里

有限的资源包括带宽、数据库链接、磁盘存储、CPU、内存、线程或应用程序特定资源等。例如，Web应用允许未被授权的用户来请求信息，它可能对于每一个HTTP请求发出很多数据库的查询。一个攻击者可以很容易地发送很多类似请求，这样数据库链接池很快就会被耗光，合法的用户将不能使用服务。还存在某些攻击变种瞄准的是一个特定的用户，如攻击者可通过不停地发送无效的有关某个用户的信息，导致系统拒绝该合法用户的访问；攻击者为用户请求新的密码，强迫他们访问E mail来重新获得访问；攻击者不停地使用系统给某个用户锁定的资源，导致其他用户不能使用，甚至有些Web应用会被立刻攻击下线。

防范拒绝服务攻击的一个通用的规则是将最少的资源分配给用户。对授权用户，可使用配额来限制用户在系统上可使用的资源；对未授权的用户，应避免任何不必要的对数据库或其他费时的资源的访问。及时检查Web应用的错误处理机制，确保一个错误不能影响整个应用程序的操作。

12.3 安全性策略

针对Web应用特性及其存在的安全威胁，有一些好的技术和方法可以预防或缓解这些攻击和漏洞。

12.3.1 安全性相关技术

Web应用的安全防护直接关系到Web应用的安全，采用一些强有力的安全性策略能极大地提高Web应用的安全性，极大地提高Web应用的质量。Web应用安全技术主要有如下几种方式。

1）“云安全”技术

“云安全”技术是P2P技术、网格技术和云计算技术等分布式计算技术混合发展和自然演化的结果，它融合了并行处理、网格计算和未知病毒行为判断等新兴技术和概念，通过网状的大量客户端对网络中软件行为的异常监测，获取互联网中病毒、木马、蠕虫和恶意软件等的最新信息，传送到服务器端进行自动分析和处理，再把病毒、木马、蠕虫和恶意软件等的解决方案分发到每一个客户端。

“云安全”可以是个开放性的系统，其“探针”应当与其他软件相兼容，即使用户使用不同的杀毒软件，也可以享受“云安全”系统带来的成果。

尽管“云安全”系统构建困难，但是“云安全”却给用户带来了巨大的好处。主要表现在病毒查杀能力全面提升、杀毒效率极大提升以及智能帮助用户做出正确安全决策。

目前一些安全软件公司都已提出相应的针对Web应用的客户端，这些客户端会实时检测Web应用，防范Web应用的挂马、Web应用病毒和钓鱼欺诈等恶意Web应用，全方位解决Web应用的安全隐患。“云安全”以一个开放性的安全服务平台作为基础，为第三方安全合作伙伴提供了与病毒对抗的平台支持。“云安全”既为第三方安全合作伙伴用户提供安全服务，又靠和第三方安全合作伙伴合作来建立全网防御体系。使每个用户都参与到全网防御体系中，遇到病毒也将不再是孤军奋战。

2）加密技术

数据在传输过程中有可能因攻击者或入侵者的窃听而失去保密性。加密技术是最常用的保密安全手段之一，它是对需要进行伪装的机密信息进行变换，得到另外一种看起来似乎与原有信息不相关的表示。合法用户可以从这些信息中还原出原来的机密信息，而非法用户如果试图从这些伪装后的信息中分析出原有的机密信息，要么这种分析过程根本是不可能的，要么代价过大，以至于无法进行。

加密技术不仅具有信息加密功能，而且具有数字签名、身份验证和秘密分存等功能。使用密码技术，不仅可以保证信息的机密性，而且可以保证信息的完整性和正确性，防止信息被篡改、伪造或假冒。

3）验证码技术

验证码是将一串随机产生的数字或符号，生成一幅图片，图片里加上一些干扰像素，由用户肉眼识别其中的数字或符号信息，输入到表单并提交给 Web 应用验证，验证成功后才能继续使用。验证码必须要求人眼能很容易地看出来，但对机器来说则很难识别。

Web 上有很多攻击软件，如注册机等，可以通过浏览 Web 页面并扫描表单，然后在系统上频繁注册或发送不良信息，影响 Web 应用的正常使用，还有的甚至可以通过多次不断的尝试盗取用户密码。采用验证码技术，可以区分出进行输入操作是人还是机器，阻止由机器自动执行的注册、信息发布等操作进而解决此问题。

4）认证技术

认证是指核实真实身份的过程，是防止主动攻击的重要技术之一，是一种可靠的证实被认证对象（包括人和事）是否名副其实或是否有效的过程，因此也称为鉴别或验证。认证可分为消息认证与身份认证。消息认证用于保证信息的完整性与抗抵赖性，身份认证则用于鉴别用户身份。在网上商务活动日益活跃的今天，很多情况下，用户并不要求信息保密，只需确认 Web 服务器或在线用户不是假冒的，自己与他们交换的信息未被第三方修改或伪造，且网上通信是安全的。

认证技术以密码技术为基础，提供了真假辨别机制，即弄清楚对象是谁，具有什么样的特征（特征具有唯一性）。认证可以是某个个人、某个机构代理、某个软件（如股票交易系统），通过一定的手段在网络上实现的过程，这样可以确定对象的真实性，防止假冒、篡改等行为。

认证技术一般又包括身份认证、公开密钥证明、报文认证、访问授权和数字签名等。身份验证机制提供判明和确认通信双方真实身份的方法，作为访问控制的基础；公开密钥证明机制对密钥进行验证；报文认证主要是通信双方对通信的内容进行验证，以保证报文由确认的发送方产生，报文传到了要发给的接收方，传送中报文没有被修改过；访问授权主要是确认用户对某资源的访问权限；而数字签名机制提供一种鉴别方法。

5）访问控制技术

访问控制是在身份认证的基础上，根据身份规定了主体对客体访问限制，是针对越权使用资源的防范（控制）措施。它对资源访问的请求加以控制，防止网络资源被非法使用、非法访问和不慎操作而造成破坏，是网络安全防范和保护的主要策略，它也是维护网络系统安全，保护网络资源的重要手段，是计算机系统中最重要和最基础的安全机制之一。

Web 应用拥有各种资源，可以是被调用的程序、进程，要存取的数据、信息，要访问的文

件、系统，或者是各种各样的网络设备，如打印机、硬盘，等等。访问控制技术是保护 Web 应用的资源受控而合法地使用的重要措施。

实现访问控制的策略主要有三种：自主访问控制（Discretionary Access Control，DAC）、强制访问控制（Mandatory Access Control，MAC）和基于角色的访问控制（Role Based Access Control，RBAC）。DAC 和 MAC 主要是采用直接对用户授予权限或取消权限。如果用户数量大而且关系复杂时，RBAC 更合适。

6）防火墙技术

防火墙是位于两个或多个网络间实施网间访问控制的一组组件的集合，它在内部与外部网络之间构筑一个屏障，由软件和硬件设备组合而成，通常处于 Intranet 与 Internet 之间，限制 Internet 用户对 Intranet 的访问及管理内部用户访问 Internet 的权限，而允许那些授权的数据通过，以阻挡外部网络或黑客的恶意攻击。防火墙被认为是安全和可信的内部网络和一个存在安全隐患的外部网络之间的封锁工具，是一种被动的技术，假设了网络边界的存在，对内部的非法访问难以有效地控制。由于防火墙能过滤不安全的服务和应用协议，从而能降低风险，极大地提高网络的安全性。

7）网络隔离技术

网络隔离（Network Isolation）技术是近些年来出现的防止非法入侵，阻挡网络攻击的一种简单而行之有效的手段。它指把两个或两个以上可路由的网络，通过不可路由的协议（如 IPX/SPX、NetBEUI 等）进行数据交换而达到隔离目的。由于采用了不同的协议，因此也被称为协议隔离（Protocol Isolation）。

8）入侵检测技术和入侵防御技术

入侵检测系统（Intrusion Detection System，IDS）是进行入侵检测的软件与硬件的组合，一般部署在有安全风险的地方，如网络出入口、远程接入服务器等，侦听网络数据流，寻找网络违规模式和未授权的网络访问尝试。当发现网络违规行为和未授权的网络访问时，入侵检测系统能够根据系统安全策略做出反应，包括实时报警、事件登录，或执行用户自定义的安全策略，等等。

但随着网络攻击技术的不断提高和网络安全漏洞的不断发现，传统防火墙技术加传统 IDS 的技术，已经无法应对某些安全威胁。因此，入侵防御系统（Intrusion Preuention System，IPS）应运而生。入侵防御系统可以深度感知并检测流经的数据流量，对恶意报文进行丢弃以阻断攻击，对滥用报文进行限流以保护网络带宽资源。

入侵检测系统的核心价值在于通过对全网信息的分析，了解信息系统的安全状况，进而指导信息系统安全建设目标以及安全策略的确立和调整；而入侵防御系统的核心价值在于安全策略的实施——对黑客行为的阻击。入侵检测系统需要部署在网络内部，监控范围可以覆盖整个子网，包括来自外部的数据以及内部终端之间传输的数据；入侵防御系统则必须部署在网络边界，抵御来自外部的入侵，对内部攻击行为无能为力。

9）防病毒技术

在 Web 应用结构中，Web 服务器作为网络的核心，为资源共享和信息交换提供了便利条件，但也给病毒的传播和扩散以可乘之机。因此，应在 Web 服务器抗病毒方面加强保护，安装服务器端杀毒软件，以提供对病毒的检测、清除、免疫和对抗能力。另外，客户端也应安装杀毒软件，将病毒在本地清除，不致扩散到其他主机或服务器上。杀毒软件并非万能的，

有时候会对操作系统进行致命的损害。所以要选择一种适合的杀毒软件，由单机防毒到网络防毒，再加上防病毒制度与措施，构成一套完整的防病毒体系。

10）数据备份与恢复技术

数据备份与恢复是使用较低廉的存储介质，定期将系统数据备份下来，以保证数据意外丢失时能尽快恢复，将损失降至最低。目前，利用磁带库或磁盘阵列是各 Web 应用通常采用的数据保护措施。

11）VPN 技术

VPN（Virtual Private Network，虚拟专用网络）是目前解决信息安全问题的最新、最成功的技术课题之一，是利用密码技术和访问控制技术在公共网络上建立专用通信网络，使数据通过安全的"加密管道"在公共网络中传播。在公共通信网络上构建 VPN 有两种主流的机制，这两种机制为路由过滤技术和隧道技术。目前 VPN 主要采用了如下 4 项技术来保障安全：隧道技术、加解密技术、密钥管理技术，以及使用者与设备身份认证技术。

12）安全脆弱性扫描技术

安全脆弱性扫描技术是能针对网络分析系统当前的设置和防御手段，指出系统存在或潜在的安全漏洞，以改进系统对网络入侵的防御能力的一种安全技术。

13）网络数据存储、备份及容灾规划

网络数据存储、备份与容灾规划是当系统或设备不幸遇到灾难后就可以迅速地恢复数据，使整个系统在最短的时间内重新投入正常运行的一种安全技术方案。

除上述 13 种 Web 安全技术外，其他安全技术还有物理安全技术、虚拟网络技术、漏洞扫描技术、主机防护技术、安全评估技术、安全审计技术、加强行政管理、完善规章制度、严格人员选任和法律介入等。但是没有一种安全技术可以完美解决 Web 上的所有安全问题，各种安全技术必须相互关联，相互补充，形成 Web 安全的立体纵深、多层次防御体系。

12.3.2 安全生命周期体系

Web 应用安全是安全教育、安全技术、安全制度的综合，是一个持续演进的过程，而不只是仅仅局限于技术领域。

Web 应用开发生命周期包括可行性分析与开发项计划、需求分析、分析设计、编码实现、测试以及发布和维护，开始得到一个初始的版本，这个过程又重新整理新的需求，根据用户反馈，新的需求又被更新、设计，加到新的开发计划里。这是一个迭代的过程。在每一个步骤都要实施安全教育，并且使用相关的安全技术，来保证常见的 Web 缺陷能够在成本最低的阶段把它解决掉。

1. 定义角色、职责

一个典型的结构包括管理决策层负责安全制度的决策，然后签署相关的政策文件。专业的安全部门负责安全技术的开发、研发，然后还有架构的安全审核、安全流程开发、漏洞的响应，这是一个专业的团队。设立一个安全专员负责整个项目，整个流程里面所有的这些安全需求、安全规范，安全专员为业务部门和安全部门起到一个沟通作用。通过这样一个结构，每个人担当什么角色都明确定义出来。这样每个人就能知道具体做哪些事情。角色和职责确定之后，就是安全意识的培养。

2. 在需求阶段就必须把安全需求提出来

为了解决 Web 应用中存在的安全隐患，从大的方面必须考虑到以下几方面的安全需求。

1）增加机密性

机密性可保护被传输的数据免受被动攻击，对于消息内容的析出，能够确定几个层次的保护。最粗粒度的方式可包含在一段时间内两个用户之间传输的所有用户数据，例如如果在两个系统之间建立一个连接，这种通用的保密服务将防止会话的所有数据的泄密。也能够定义这种服务较细粒度的形式，包括保护单一消息中的某个特定字段。细粒度方式实现起来更为复杂且成本更高。机密性的另一个方面是包含通信量免受分析，这要求一个攻击不能够在通信设施上观察到通信量的源和目的、频度、长度等特征。机密性服务主要是通过加密的方法来实现的，它的可靠性取决于加密的算法密钥的长度以及密钥管理等方面。

2）提供身份鉴别

身份鉴别的目的在于确保一个通信是可信的。在诸如产生一个警告或警告信号的单个消息的情况下，鉴别服务的功能是能向接收方保证该消息确实来自它宣称的源。在诸如一个客户端与服务端连接这样一个正在进行的交互情况，身份鉴别服务涉及两个方面：首先，在连接发起时，服务应确保这两个实体是可信的；其次，该服务必须能够确保该连接不被干扰，使得第三方不能加密这两个会话方中的任何一个来达到不经授权传输或接收数据的目的。身份鉴别服务可以通过用户名、密码的方式来实现，但这种方式的可靠性不强，目前广泛使用的是身份证书来进行主体的身份鉴别，其安全性主要取决于 CA 的可信度。

3）保证消息的完整性

如同机密性一样，完整性能够应用于一个消息流、单个消息或一个消息中的所选字段。同样，最为有用和直接的方法是对整个流的保护。一个面向连接的完整性服务是处理消息流的服务，它能确保接收到的消息如同发送的消息一样，没有冗余、插入、篡改、重排序或延迟，该服务也包括数据的销毁。因此，面向连接完整性服务用于处理消息流篡改和拒绝服务。在另一方面无连接完整性服务用于处理任何没有较长内容的单个消息，通常只保护免受篡改。对需要免受重放和重排序的某些保护而不需要严格排序的某些应用程序而言，能够提供一个混合服务。数字签名对发送的信息进行散列后得到该信息的定长的特征后，通过数字证书签名就可以保证该信息是完整的、没有受到修改的，保证完整性，增强消息的不可抵赖性。

4）保护服务的可用性

各种攻击能够导致可用性的丧失或减少，例如拒绝服务攻击常常针对系统的高性能代价的服务，导致系统服务能力下降。保护服务的可用性方面通过身份的鉴别等措施来尽量保证用户的可信度，另一方面通过给系统提供审计功能，对于各类可识别的攻击手段采取应对措施。提供网络安全环境中的访问控制，限制和控制通过信道对主机系统和应用程序进行访问的能力。为了取得这种控制，每个试图得到访问的实体必须先进行身份识别或被鉴别，系统依据对不同用户访问权限的限制来确保资源和服务的安全性。

3. 设计一个安全的架构

明确资源、角色、边界之后就可以对不同的资源提出不同的安全需求，提供不同程度的保护措施。当这些都确定后，可以设计一个最简单的方案，包括安全的措施，并且在这个过程中设计一个验证的方案和风险分析。这个主要通过几个步骤来实现，首先是需要分析运营环境中有哪些安全因素，是网络这一层还是应用这一层，还是合作方这一层。确定最有可能的威胁，哪一些威胁是最严重的、最有可能发生的，一旦发生之后它的影响会有多大，它发生的频率有多高。一旦这些清楚之后，就可以制定降低威胁和风险的措施。当这个架构设计好之后，通常应该经过一个验证、评审阶段。还有在架构进行重大调整的时候，也需要评审，看这个重构会不会对现有架构产生新的威胁。

4. 安全编码

对于跨站点、跳转和注入式攻击等漏洞，大多数都是非常简单的不安全编码疏忽所导致的，可以在编码这个阶段通过使用一些安全编码实践来避免这些问题。在这个阶段可以提供一些工具的支持，比如静态代码分析，可以分析一些SQL注入、跨站脚本，同时可以提供一些脆弱性测试工具。在这个阶段还要进行重审代码，把残留的不安全代码清理掉。

5. 安全测试

在测试阶段为安全需求创建测试策略和用例，除了常规测试外，专门针对安全性进行测试。

6. 发布和维护阶段

部署之前要确保运营主机环境安全，主要是一些配置。通过在线监控和扫描可以发现存在的漏洞和文件篡改，对于一些入侵也可以进行某些告警，同时对业务异常信息进行监测与分析。

通过以上几个安全开发周期的相关策略，基本可以消灭大部分的安全隐患。但是还是可能会存在漏网之鱼，要通过安全响应来进行善后处理。经常监视这些漏洞的来源，是来自外部报告还是内部渗透测试，还是被一些地下组织利用。漏洞发现之后要分析与定位，到底涉及哪些系统，发现的问题是在哪个地方，然后制定修复计划，进行回归测试，确保没有问题之后再重新发布上线。

12.4 客户端安全防护

客户端安全是整个Web应用安全不可分割的一部分，绝大多数用户是以客户端的身份出现在网络上，在客户端输入的敏感信息(如银行卡的账号、密码等)一旦被黑客或不法分子非法窃取，那么无论是对个人还是对服务提供商，都是巨大的损失。客户端安全离普通用户的生活更为贴近，也更为现实，更加容易受到黑客以及不法分子的攻击。

面对病毒、木马、蠕虫以及黑客攻击手段紧密结合的现状，做好客户端的安全防护，首先

应该在客户端安装杀毒软件、防木马软件等应用软件。并保持即时更新，开启入侵防护系统病毒木马防护策略，建立统一病毒防护体系，如在网络边界部署网络防毒墙，对关键服务器和客户端进行重点防护，并对其网络连接进行入侵检测监控，保证安全风险在一个可控的范围内。

客户端所涉及到的 Cookie、JavaScript、ActiveX 控件、Java Applet、插件等技术都存在一些安全隐患（如 12.2.2 节所述），需要综合考虑以防范可能出现的安全漏洞。如对 Cookie 的属性进行设置，使其只能在使用 SSL（安全套接字）的连接上传输，防止其在传输途中被他人截取；针对 JavaScript 对 Web 应用进行充分的测试；ActiveX 通过验证码（Authenticode）进行代码签名；对 Java Applet 进行签名，设置细粒度的访问控制；安装插件时要确认其来源可信。

在客户端安装杀毒软件、防火墙，配置好客户端的浏览器设置，并不代表 Web 应用就安全了。新的安全漏洞不断涌现，新型病毒、木马等也不断产生，新的攻击不断发生，对于这些不断出现的安全威胁，客户端安全的防护还需要有一些良好的管理制度，从用户的操作行为入手加以管理。

对用户而言，首先需要加强安全防护意识，同时还要学会防范黑客的恶意攻击，掌握互联网上的防攻击策略。以下是用户需要注意的操作事项。

(1) 用户尽量不要随意在 Web 应用上填写真实信息，很多假冒的 Web 应用都是通过这种方式窃取用户信息的。

(2) 不要随意相信和打开陌生邮件，正规 Web 应用一般不会索要用户的资料，而会让用户登录后做相关操作。

(3) 不要打开陌生人发送的文件和链接，许多恶意代码也会巧妙地隐藏在一些 Web 页面之中。

(4) 不要随意访问陌生的 Web 应用，对于不了解的 Web 应用，随意访问是一种很危险的举动，这些 Web 应用会强制修改用户的浏览器设置。

(5) 不要轻易安装来历不明、用户评价度不高的应用软件，以避免捆绑有木马或是恶意插件的软件被安装。

(6) 重要邮件要加密。

市场上出现的客户端安全方案集成了可集中管理的防病毒、个人防火墙和入侵检测等安全模块，缩短了用户在混合型病毒发作时的关键响应时间。这类将各种安全防护手段整合在一起的方案，既为用户节约了采购及维护成本，又极大地提高了客户端的安全防护水平，做到了全方位防护。

总之，用户一般很容易忽视客户端安全防护，而客户端安全直接与用户相关，所以一定要加强对这方面的管理，从客户端建立起一道全面、立体的安全防线。

12.5 服务器端安全防护

服务器端病毒防护与客户端防护有许多共同之处，二者都试图保护同一基本个人计算机环境。服务器是 Web 应用的灵魂，服务器端的安全问题较之客户端而言有过之而无不及。两者的主要差异在于，服务器端防护在可靠性和性能方面的预期级别通常高得多。此外，鉴于许多服务器在组织基础结构中起到的特有作用，通常需要制定专门的防护解决方案。

虽然已有防火墙和入侵检测等安全防范手段，但各类Web应用系统的复杂性和多样性致使系统漏洞层出不穷、防不胜防。据中国被黑站点统计系统数据分析，截止到2010年12月31日24时，2010年登记在案的被黑网站数量已达18 248个(其中有很多网站是多次被篡改)，平均每天约有50个站点被篡改且被举报。针对这些情况，迫切需要服务端的网页防篡改措施，由此产生了网页防篡改系统。到目前为止，网页防篡改技术主要可分为三类。

1）时间轮询技术

时间轮询技术是利用网页检测程序，以轮询方式读出要监控的网页，与真实网页相比较，来判断网页内容的完整性，对于被篡改的网页进行报警和恢复。但是，采用时间轮询式的网页防篡改系统，对每个网页来说，轮询扫描存在着时间间隔，在间隔中，黑客可以攻击系统并使访问者访问到被篡改的网页。

2）核心内嵌技术结合事件触发技术

核心内嵌技术即密码水印技术，最初先将网页内容采取非对称加密存放，在外来访问请求时，将经过加密验证的进行解密对外发布，若未经过验证，则拒绝对外发布，调用备份网站文件进行验证解密后对外发布。此种技术通常要结合事件触发机制对文件的部分属性进行对比，其最大特点是安全性相对时间轮询技术有了很大提高，不足是加密计算会占用大量服务器资源，影响系统的反应速度。

3）文件过滤驱动技术结合事件触发技术

其原理是将篡改监测的核心程序通过文件底层驱动应用到Web服务器中，通过事件触发方式进行自动监测，对文件夹的所有文件内容，对照其底层文件属性，经过内置散列快速算法，实时进行监测。若发现属性变更，通过非协议方式、纯文件安全复制方式将备份路径文件夹内容复制到监测文件夹相应文件位置。但该技术需要依赖特定的文件系统。

12.6 客户-服务器之间通信防护

Web浏览器和Web服务器之间的信息交换是通过数据包的网络传输来实现的，所以Web数据传输过程的安全性直接影响着Web应用的安全。

1. 安全服务分类

网络通信安全是指网络信息的机密性、完整性、可用性、真实性、实用性和占有性，不仅包括计算机内信息存储的安全性，还包括信息传输过程的安全性、网络结点处的安全和通信链路上的安全。网络通信安全就是拟定一定安全策略，使信息在网络环境中的保密性、完整性及可用性受到保护。

(1) 机密性。确保在计算机网络中的数据和被传输的数据仅能被授权允许读取的各方得到，未授权的用户不能访问系统中的数据。

(2) 鉴别。确保一个消息的来源或电子文档被正确地标识，同时确保该标识没有被伪造。

(3) 完整性。确保数据正确、有效和一致，不因为人的因素而改变数据原有内容、形式与流向，即不能被未授权的第三方修改(包括对传输数据的写、改、改变状态、删除、创建、时延或者重放)。

(4) 可用性。可用性要求计算机系统的资源在需要时可被授权各方使用，即保障数据

资源能提供既定的功能，无论何时何地，只要有需要即可服务，而不因系统故障、误操作等造成资源丢失，妨碍对资源的使用，服务未能及时响应。

(5) 不可抵赖。无论发送方或者接收方都不能抵赖所传输的数据。

(6) 访问控制。访问控制要求对数据源的访问可由目标系统控制。

网络数据传输过程中主要面临被动攻击和主动攻击两方面的威胁。被动攻击主要包括流量分析，监视和窃听明文，解密弱加密的数据流，获取鉴别信息(如口令)，等等。攻击者利用这些信息可能窃取传输的信息，分析、判断网络的规模、拓扑结构，定位结点位置、设备特性，等等。主动攻击主要包括修改传输数据，重放、劫持会话，伪装成授权的用户服务器、未授权使用应用程序的操作系统软件，利用数据执行、插入和利用恶意代码(如木马、后门、蠕虫)，利用协议或基础设施的错误，拒绝服务。攻击者利用篡改或伪造的信息插入网络，尤其是对网管和信令信息的攻击，会影响网络设备的正常运行，甚至造成网络配置混乱，使整个网络瘫痪。

2. 网络通信安全主要内容

(1) 典型的网络威胁，主要有恶意攻击、安全缺陷、软件漏洞和结构隐患等。

(2) 安全机制和服务，主要有认证、保密、数据完整性和访问控制等。

(3) 网络安全协议，主要包括如下几种。网络层协议：IPSec(包括认证头协议 AH，封装安全载荷协议 ESP)。安全套接层(介于传输层和应用层之间)协议：SSL。应用层协议：安全电子交易(SET)协议、安全多用途 Internet 邮件扩展和 PGP 加密等。

(4) 安全产品种类，它是各生产厂商制作好的有特定安全功能的产品，主要种类有防火墙、VPN(虚拟专用网)、扫描器、入侵检测系统、SVN(安全虚拟网络结构)、密码机、防毒软件、安全网关、网络流量分析产品以及各种各样的整体解决方案等。

3. 网络通信安全技术

计算机网络安全机制必须在保障计算机网络可靠性的前提下，保证计算机网络中的数据的机密性、完整性、可用性、可控性和不可抵赖性。由于网络需要在任何时间能为分布于不同地域和不同组织的用户提供便捷和快速的数据传输通道，因此，采用如下适当的安全策略，建立安全可靠的计算机网络已成为迫切要求。

1) 防火墙技术

防火墙技术是一种隔离技术，是在两个网络通信时执行的一种访问控制，它能最大限度地阻止网络中的黑客更改、复制、毁坏重要信息。

2) 数据加密技术和用户授权访问控制技术

为了确保数据不会在设备上或传输过程中被非法窃取，对网络传输数据加密是一个十分重要的安全措施。数据加密技术分为数据传输、数据存储、数据完整性鉴别和密匙管理技术等。数据传输加密技术能够准确判别该输入是否来自合法用户，而用户授权访问控制技术主要实现密匙管理，通过对密匙的产生、存储、传递、定期更换进行有效的控制，实现网络通信信息的安全。

3) 认证和身份鉴别机制

为了使网络具有是否允许用户存取数据的判别能力，避免出现非法传送、复制或篡改数

据等不安全现象，网络需要采用认证和身份鉴别机制。认证是网络安全的基本机制，网络设备之间必须互相认证对方身份，才能保证正确赋予操作权力和数据的存取控制，确保数据的机密性。数据传输系统的安全性往往取决于能否正确识别用户或终端的身份。目前用于身份认证的主要技术有：验证用户知道什么（如口令、密钥等），验证用户拥有什么（如钥匙、徽标、IC卡等）、验证用户的生理特征（如指纹、声音等），验证用户的习惯动作（如笔迹等）。

4）审计

审计是防止内部犯罪和事故后调查取证的基础。通过对一些重要的事件进行记录，从而在系统发现错误或受到攻击时能准确定位错误和发现攻击。审计信息必须具备防止非法删除和修改的措施。具体来讲，审计主要是为了达到以下几个方面的目的。

（1）辅助辨识和分析未经授权的活动或攻击。

（2）帮助并且保证那些实体响应行动处理这些活动。

（3）促进开发改进的损伤控制处理程序。

（4）认可与已经建立的安全策略的一致性。

（5）报告那些可能与系统控制不相适应的信息。

（6）辨识可能需要的对控制、策略和处理程序的改变。

5）数据完整性保护

数据完整性是指数据在存储或传输过程中不被非法修改、破坏或丢失。首先，网络会充分利用数据库管理系统提供的数据完整性的约束机制和各种输入数据的"引用完整性约束"设计，确保数据完整、准确地输入和储存。其次，网络在数据传输过程中可视情况选用相应的数据校验方式对传输数据进行校验检查，如发送方在发送的消息中加入鉴别码并经加密后发送给接收者。完整性的另一功能是提供不可抵赖服务。当数据源的完整性可以被验证却无法模仿时，接收方可以认定数据的发送者，数字签名就可以提供这种手段。

6）数字签名

数字签名是一种类似写在纸上的普通的物理签名，但是使用了公钥加密领域的技术实现，是一种用于鉴别数字信息的方法。

数字签名技术是不对称加密算法的典型应用，它的应用过程如下：发送数据时，将发送的数据采用传统的加密方法（如DES算法）得到的密文和用来解码的密钥一起发送，但发送的密钥本身必须用公开密钥密码算法的公开密钥PK加密，到目的地后先令一个密钥SK来解开传统加密方法中的密钥，再用该密钥解开密文。

7）权利管理与访问控制

权利管理和访问控制是系统必备的安全手段。系统根据合法的认证，赋予某用户适当的操作权限，却不能进行越权的操作。权利管理和访问控制一般采用角色管理办法，针对系统需要定义各种角色，如会计、财务等，然后赋予他们不同的执行权利。

4. Web安全协议

Web安全协议广泛地用在Internet和Intranet的服务器产品和客户端产品中，用于安全地传送数据，并将数据集中到每个Web服务器和浏览器中，从而来保证用户与Web应用安全交流。

1) SSL

SSL(Secure Socket Layer,安全套接字协议)是 Netscape 所研发的在 Internet 基础上提供的一种保证私密性的安全协议,它利用数据加密技术,可确保数据在网络传输过程中不会被截取及窃听。

SSL 协议要求建立在可靠的传输层协议(如 TCP)之上,SSL 协议的优势是其与应用层协议无关,高层应用层协议(如 HTTP、FTP、Telnet 等)能透明地建立于 SSL 协议之上。SSL 协议在应用层协议通信之前完成加密算法、通信密钥的协商以及服务器认证工作。此后,应用层协议所传送的数据都会被加密,从而保证通信的私密性。

HTTPS 是以安全为目标的 HTTP 通道,简单来说就是 HTTP 的安全版,它在 HTTP 下加入 SSL 层作为其安全基础。

2) TLS

TLS(Transport Layer Security,安全传输层协议)用于在两个通信应用程序之间提供保密性和数据完整性。该协议由 TLS 记录协议(TLS Record)和 TLS 握手协议(TLS Handshake)两层组成。较低的层为 TLS 记录协议,位于某个可靠的传输协议(如 TCP)上面。

TLS 的优势在于它是独立的应用协议,高层协议可以分布在 TLS 协议上面。然而 TLS 标准把如何添加 TLS 协议安全性、如何启动 TLS 握手协议以及如何解释认证证明的决定权留给运行在 TLS 上的协议的设计者和实施者来判断。

3) IPSec

IPSec(IP Security)协议是 Internet 工程任务组为 IP 安全推荐的一个协议,通过相应的隧道技术,可以实现 VPN。IPSec 协议组还包括支持网络层安全性密钥管理要求的密码技术。ISAKMP(Internet Security Association Key Management Protocol,Internet 安全协定密钥管理协议)为 Internet 密钥管理提供框架结构,为安全属性的协商提供协议支持。它本身不能建立会话密钥,但它可以与各种会话密钥建立协议一起使用,为 Internet 密钥管理提供完整的解决方案,如 Qakley。

IPSec 工作时,首先两端的网络设备必须就安全关联(Security Association,SA)达成一致,这是两者之间的一项安全策略协定。SA 包括加密算法、鉴别算法、共享会话算法、密钥使用期限。SA 是单向的,故欲进行双向通信需要建立两个 SA(各为一个方向)。这些 SA 通过 ISAKMP 协商或人工定义。SA 商定之后,再确定是使用鉴别、保密和完整性或仅仅只用鉴别。

IPSec 有隧道和传输两种模式。隧道模式下,整个 IP 数据报、IP 报头和数据都封装在 ESP 报头中。在传输模式下,有数据部分是封装,而 IP 报头则不封装即被传送。目前,标准规定必须实施密码块链接(CBC)模式中的 DES。

IPSec 接收端的网络设备根据接收端的 SA 数据库对使用 IPSec 加密的数据进行相应的解密并接收,以达到传送数据的私有性和完整性。

12.7 总结与展望

功能和性能,往往是衡量应用是否满足需求的指标。然而,对于以 Internet 为基础架构的 Web 应用而言,安全性是尤为重要的考量标准。即使功能再完备、性能再可靠的 Web 应

用，一旦遭到黑客的攻击和破坏，一切都失去意义。因此在构建Web应用时，需要加强对应用安全的重视程度。

本章从Web应用安全性特性和Web应用所面临的安全威胁出发，阐述了Web应用的安全性策略，以及客户端安全防护、服务器端安全防护以及通信安全和Web安全协议。

但从目前现状来说，因特网上的Web应用大都存在着极大的安全隐患和风险，需要从网络、Web服务器以及程序安全三方面着手以加强Web应用的安全。网络方面，考虑将防火墙、IDS IPS、安全网关、防病毒墙和网站保护墙等部署在Web服务器前面；Web服务器方面，通过部署更安全的Web服务器、Web服务器自身的保护（如网页防篡改保护、恶意主动防御、访问控制和审计）系统等来保护系统和文件；程序方面，考虑整个Web应用开发生命周期，将安全意识贯彻到开发的每个阶段的每个人上，及时发现和排除应用中存在的安全隐患。

第13章 Web工程的发展

随着互联网技术的飞速发展，短短的十几年时间，Web已经从信息共享，发展成各类复杂的和混合的Web应用，并融入到人们的工作与生活中。信息种类和信息量呈爆炸式增长，Web应用使用的范围越来越广，上下文环境不断演化。如果说这十几年告诉我们什么，那就是Web的发展速度快得惊人，影响大得无法形容，而且每个Web应用又具有其独特的特性。

Web发展过程中技术和标准在快速更新，工具在随之演化，更好的方法也在不断出现。那么，未来又如何发展，提供什么样的用户交互，如何改变人们的生活，会有哪些工具和技术帮助Web工程师应对Web的发展呢？一方面，Web和Web应用的特征不断变化，其架构不断演化；另一方面，计算机程序对信息的智能推理与理解能力仍然匮乏。W3C明确指出：语义Web技术能为用户提供高水平的信息服务，是最好、最先进的技术，Web的未来就是语义Web。即未来的计算机主要是自动地发现、聚合、组织、分析、解释和推理等。Web工程的发展正是要实现计算机的分析和理解能力，改进Web工程过程，认识解决方案在开发过程中的演化，适应业务环境的动态特性并控制变更。

13.1 Web技术的演化

Web从十多年前的信息传播工具，已经演化为一种改变人们生活方式的Web应用环境，为人们提供了平等交流平台。人们可以通过Web发表自己的观点，记录生活中的点点滴滴。Web已经成为一个基于用户关系的信息分享、传播和获取平台，用户可以组建个人社区，如博客(Blog)、微博(MicroBlog，如Twitter)、WiKi(如Wikipedia)、社会网(Social Networking，如Facebook、Flickr和人人网)等。HTML5的技术性突破，又可以将桌面应用程序功能带入浏览器中。综观Web发展历程，人们把Web的发展经历归纳为Web 1.0、Web 2.0、Web 3.0等不同的阶段，如图13.1所示。Web 1.0代表了第一代互联网，主要是网站中心化。下面从Web 2.0开始介绍。

13.1.1 Web 2.0

关于Web 2.0的较为经典的定义是Blogger Don在《Web 2.0概念诠释》一文中提出的："Web 2.0是以Flickr、Craigslist、Linkedin、Tribes、Ryze、Friendster、Del. icio. us、3Thing. com等站点为代表，以Blog、Tag、SNS、RSS、WiKi等社会软件应用为核心，依据六

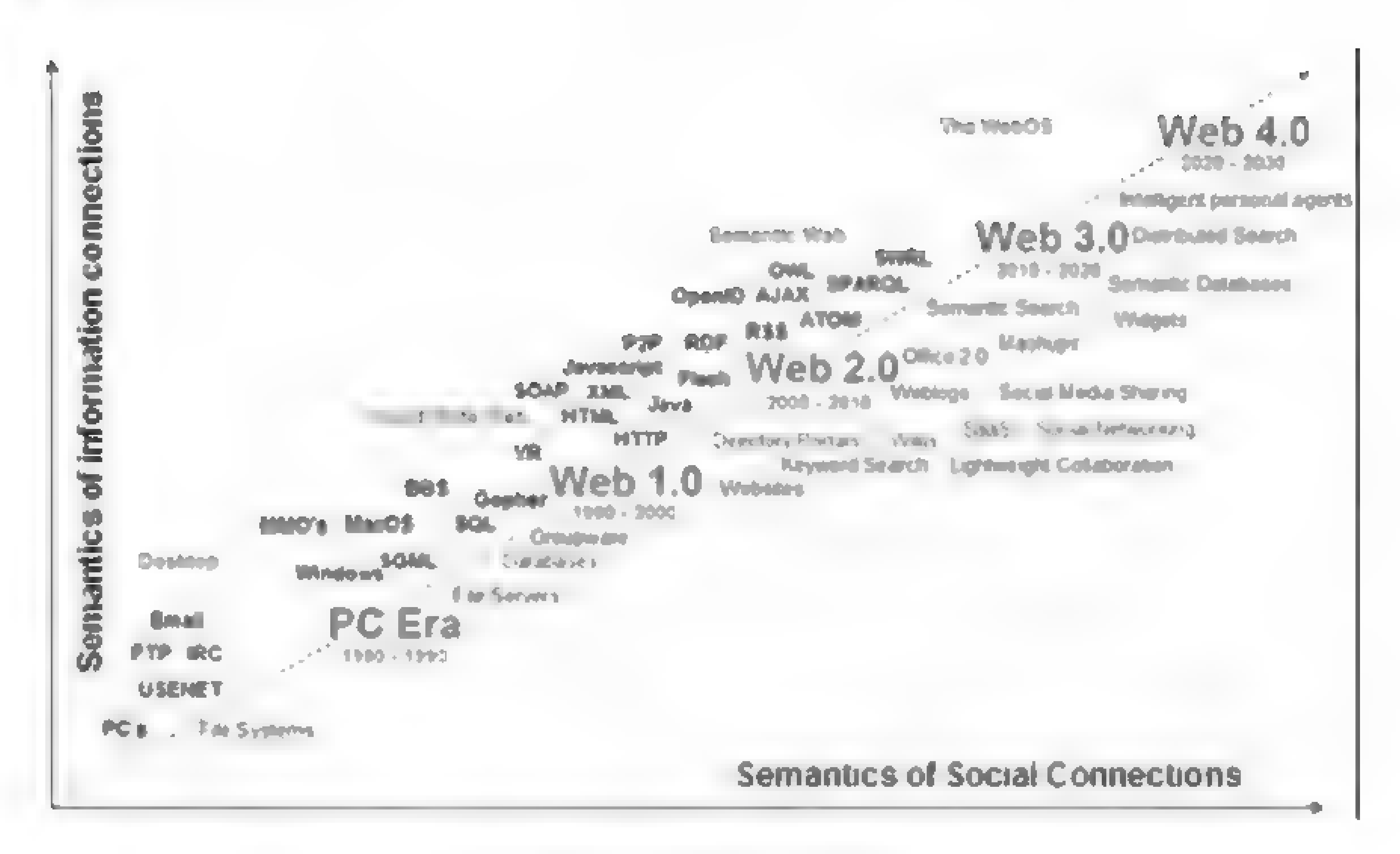


图 13.1 Web 发展历程

度分割、XML、AJAX 等新理论和技术实现的互联网新一代模式。Web 2.0，是相对于 Web 1.0 的新的一类互联网应用的统称，是一次从核心内容到外部应用的革命。"从 Web 1.0 到 Web 2.0 的发展主要体现在模式、基本构成单元、工具和内容创建者 4 个方面。两者的主要区别如表 13.1 所示。

表 13.1 Web 1.0 与 Web 2.0 的对照

	Web 1.0	Web 2.0
发展时间	1993—2003 年	2003 年以后
模式	读	写、共同建设
基本构成	页面	发表和(或)记录的信息
工具	互联网浏览器	各类浏览器、RSS 阅读器
运行机制	客户端，服务器	Web 服务：新兴的应用，如博客、WiKi、分类广告、社会书签、交友站点等，传统的应用如邮件列表、BBS、新闻组
作者	程序员等专业人士	全部普通用户
应用	初级应用	全面大量应用

Web 2.0 并不是指某种(组)特定的技术，而是指 Web 应用设计的趋势，是一种新现象，一种理念，描述 Web 应用站点和技术的无缝结合，以改进传统站点的一维特性。Web 2.0 回到了 Tim Berners-Lee 最初的原理，使得 Web 变得更加具有参与性，即交互性，追求好的用户体验。如 Google Maps 将卫星信息和航拍图片结合；Facebook 提供好友之间的动态信息交换，提供基于浏览器的游戏；等等。几乎所有的社会网工具(如 Blog 和 WiKi 等)都是归属于 Web 2.0。

Web 2.0 也包含了技术架构及应用软件(技术体系如图 13.2 所示)，它的特点是鼓励信息的最终利用者分享资源，如社会网、Blog、WiKi、AJAX、Mashup 等。Web 2.0 也提供了新的业务模式，如广告条转变为，针对不同兴趣爱好的社会网络群体，投放不同分类广告的主

动推选，甚至可以主动收集人们的网络使用习惯、偏好等，推送完全不同的个性化服务，也相应地引出了“Blog 营销”、“长尾现象”等新经济模式。

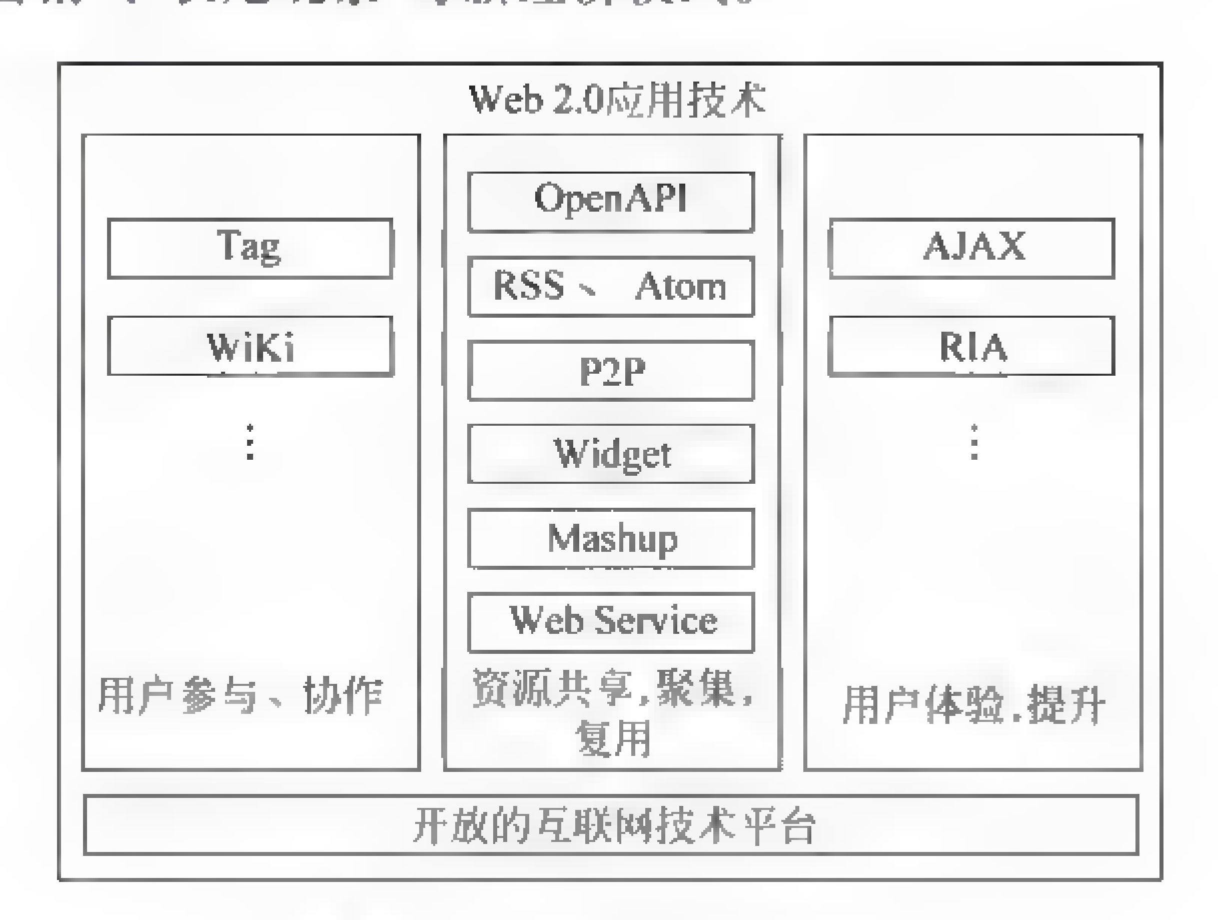


图 13.2 Web 2.0 技术体系

总的来说，Web 2.0 具有如下几个方面的特性。

(1) 以人为中心。Web 2.0 以个人为中心，人是灵魂，鼓励多人参与，发表个人看法。个人深度参与到互联网中，也就是说用户每个人都既是信息的生产者，又是信息的消费者，如标签 Tag、多媒体、在线协作等。通过 Blog，不需要任何的专业知识，就可以在 Web 上发布信息。

(2) 既可读又可写。Web 2.0 是“可写可读互联网”。

(3) 服务众多。Web 2.0 包含了人们经常使用到的服务，例如博客、播客(Podcast)、维基、P2P 下载、社区、分享服务等。

(4) 真实性。Web 2.0 时代的一项基本的原则就是真实，如果要隐藏身份，也就失去了利用 Web 2.0 的便利性，比如用户不能把自己的照片共享，也不能让好友方便地找到，而用户所发布的信息的可信度也会大打折扣。

(5) 聚合性。Web 2.0 的首要的也是最重要的发展是使用标准化协议的站点之间内容的联合，使最终用户在其他环境中使用站点的数据，包括另一个站点、浏览器插件或者一个单独的桌面应用程序，这些联合协议包括 RSS、RDF(资源描述框架)和 Atom。采用的协议如 XML、FOAF 和 XFN(XHTML 朋友网络)。

13.1.2 Web 3.0

Felice Bochman 从 Web 3.0 会议中概括出“Web 3.0 is about user-experience, meaning, intent, story, relevance, mobile, data, share, and free content as a service.”，并将其总结为一个词“real(真实)”。Google CEO 埃里克·施密特认为，Web 3.0 是组合在一起的应用程序，即互联网将由一系列标准的 Web 组件拼装在一起。这些应用程序虽单体相对较小，但运行速度非常快，并能进行很多自定义，可以在任何设备上运行(如个人计算机或移动电话)，而且由于数据处于云中，它可以像病毒一样扩散(如社交网络和电子邮件等)。也有人认为 Web 3.0 主要用于概括 Web 发展过程中可能出现的各种不同的方向和特征，

包括数据 Web，跨浏览器、超浏览器的内容交付和请求机制，人工智能技术的运用，语义 Web，地理映射网，3D 虚拟世界或网络公园，等等。

如表 13.2 所示，Web 3.0 相比 Web 2.0，更加强调"智能化的机器与机器之间的交流，机器与人之间的交流"的互联网模式，提供基于用户偏好的个性化聚合服务。例如，使用智能搜索引擎可以接受复杂句子作为搜索数据，并且返回基于对句子理解的结果；甚至，互联网不仅提供信息服务，还能够提供个性化的顾问服务，比如基于人们的互联网行为轨迹，专业的 Web 应用将成为一个能针对简单问题给出合理、完全问答的系统，将 RSS(Really Simple Syndication，简易信息聚合)的应用提升到另一个应用层面；交互式 3D 内容在使用 HTML5 后，可以不使用 Flash 或 Silverlight 等插件在不同浏览器和平台上使用。

表 13.2 Web 2.0 与 Web 3.0 的对照

Web 2.0	Web 3.0
"社会网"(The Social Web)	"语义网"(The Semantic Web)
丰富的信息(Information)	丰富的含意(Meaning)
社会媒体(Social Media)	智能代理(Intelligent Agents)
第二个十年：2000—2010	第三个十年：2010—2020
Google 作为催化剂	链接的数据(Linked Data)作为催化剂
搜索(Searching)	发现(Finding)
集体的智慧(Wisdom of Crowds)	语义工具的智慧(Wisdom of Semantic Tools)
Google 的页面排名(PageRank)	语义搜索引擎
HTML、SGML	HTML5、XML
无规则，关键词，混乱	标准，元数据(Metadata)，协议
打印和数字访问	产生数字化，开放式访问(OA)

1) Web 3.0 的特点

Widget(微内容，微件)的自由整合与有效聚合；适合多种终端平台，实现信息服务的普适性，如 PC 互联网到 WAP 手机、PDA、机顶盒、专用终端；良好的人性化用户体验，以及基础性的个性化配置；有效和有序的数字新技术，Web 3.0 将建立可信的 SNS，可管理的 VoIP 与 IM(即时通信)，可控的 Blog/Vlog/WiKi，实现数字通信与信息处理、网络与计算、媒体内容与业务智能、传播与管理、艺术与人文的有序有效结合和融会贯通。HTML5 将成为 Web 3.0 时代的技术基础。

2) Web 3.0 应用

Web 之父 Tim Berners-lee 提出"Web 3.0 is something called the semantic Web"。Web 3.0 在全部的信息、资源、知识等数据之间创建语义关联，形成 Linked Data，并提供基于语义的服务。Web 3.0 应用体现在将信息的互通进一步深度挖掘，直接从底层数据源进行互通，实现基于语义的检索、全民互动盈利模式以及精准营销模式。

13.1.3 Web 4.0、Web 5.0

Web 技术的发展迅速，各种新技术层出不穷，人们对此毫无质疑，因而出现了以大约 10 年为一个周期的预期发展目标：Web 4.0 和 Web 5.0。Web 4.0 的核心知识分配系统旨在解决应该如何学习知识这一问题。Web 5.0 指的就是语用网(Pragmatic Web)。

语用网针对的是技术背后的模型本质。现有的计算机技术都是图灵机模型，图灵机是机械化、程序化，或者说算术，以数据和算符(算子)为2元的闭合理论体系。图灵机是研究和定义在数据集上的算子规律或法则的数学科学。在网络世界里，这些封闭系统都要联合起来，成为一个整体，即整个网络成为一台计算机系统，而这台计算机是Petri网了。Petri网计算机的存在环境叫语用网。在语用网中，一切事务性的工程工作都由计算机系统完成，互联网的所有系统自动通信，有效协作。

13.2 Web应用的发展演化

随着计算机硬件、计算机科学、网络计算模式、移动互联网等的不断快速发展，Web技术也得到了巨大的发展，Web应用也随之经历了或面临着巨大的发展演化。诸如Mashup、Comet和Widget等新Web技术的使用，网格计算、云计算等新计算模式的出现，移动互联和各种可移动设备的不断发展，HTML5等新标准的出现，已经或将会引起Web应用的进一步发展和演化。

13.2.1 新的Web应用种类

Web本身的发展，出现了各种应用种类和相应的技术，使得Internet的应用模式也有相应的变化，例如Mashup、Comet和Widget等。

1) Mashup

Mashup应用是Web 2.0时代一种崭新的应用模式，将多个不同的Web应用源的数据、展示和(或)功能混搭成Web应用。Mashup运用Web 2.0技术，通过开放API(应用程序接口)、RSS等方式把不同内容源快速聚合起来。它与AJAX的结合增加了与用户的交互性；简单的聚合原理与图形化界面的开发工具，产生一种新的应用软件开发模式；通过对API的二次开发，Mashup能够快速简易地开发出的新的应用。比如，天气服务的开发者没有想到天气服务开发出的API会被整合进春运地图并且发挥了巨大的作用。Mashup得到了极客(geek)和Internet玩家的极大欢迎。

为了方便数据的检索，提供者通常会将自己的内容通过Web协议对外提供，如REST、Web Service和RSS ATOM。Mashup提供者还可以在数据和内容源提供者不知情的情况下通过屏幕抓取来获取相应的数据信息。

Mashup站点是Mashup逻辑所在地，但不一定是执行这些逻辑的地方。通常，Mashup使用服务器和客户端逻辑的组合来实现自己的数据集成。很多Mashup应用程序都是直接用了由用户提供的数据。另外，客户端的Web浏览器以图形化的方式呈现Mashup应用程序，同时也是用户交互发生的地方，但是一些复杂的应用逻辑不可能只在客户端的Web浏览器中执行就能完成。

Mashup所涉及的编程语言可分为三类：面向浏览器端的数据、服务集成组织语言，主要是各类脚本语言，如支持AJAX技术的JavaScript等；服务器端程序语言，实现服务器端的复杂功能和操作，主要是各类高级语言，如Java、PHP、Perl等；Mashup过程定义描述语言，如DSL。

Mashup的典型应用如地图Mashup，不同公司相继公开了自己的地图API，如Google Maps API，微软的Virtual、Yahoo Maps和AOL（MapQuest）；视频和图像Mashup，如Flickr使用其API来共享图像，CNN之类的新闻站点作为输入，并在新闻中通过照片匹配以文字形式呈现照片中的内容；搜索和购物Mashup，如Google的Froogle、MySimon等，eBay和Amazon等消费Web应用发布了自己的API，为这类Mashup应用提供了基础；新闻Mashup，如Diggdot.us，它将Digg.com、Slashdot.org和Del.icio.us上与技术有关的内容进行聚合。为了创建Mashup应用，许多公司推出了自己的Mashup平台，如Yahoo Pipes、Google Mashup Editor、Microsoft Popfly、IBM QEDWiki和Intel Mash Maker等。

2）Comet

Comet也被称为Reverse AJAX，是一种基于HTTP长连接及服务器端事件驱动架构。在这种架构的应用中，服务器端会主动以异步方式向客户端程序推送数据，而不需要客户端显式地发出请求。向浏览器端实时传送服务器产生的异步事件的Server Push（服务器推送）技术主要为现有的Web应用提供数据的实时传输，减少数据的延迟，非常适合事件驱动的Web应用，以及对交互性和实时性要求很强的应用，如现今流行的Web在线游戏（基于浏览器的）、Web在线股票信息系统、Web在线即时通信、基于浏览器的在线实时监测系统、Web在线直播等。

Comet技术所具有的三大特征如下。

（1）基于HTTP长连接技术。

（2）服务器端采用异步事件驱动架构。

（3）数据传输的主动性和实时性。

如果说AJAX技术实现了浏览器端向服务器端异步地请求数据，而Comet技术则实现了服务器端主动向浏览器端推送数据。从事件驱动的角度来说，AJAX技术是把浏览器端产生的异步事件发送给服务器，然后由服务器对事件做出响应；而Comet技术则是把服务器端产生的异步事件主动推送给浏览器，然后由浏览器对事件做出响应更新。Comet技术与AJAX技术相结合，将大大提高Web应用程序的交互性和响应性，增强用户体验。

Comet应用的实现模型一般有以下两种方式：基于AJAX的长轮询（Long-polling）方式和基于iframe及htmlfile的流（Streaming）方式。

使用AJAX实现服务器推送，与传统的AJAX应用不同之处在于：服务器端会阻塞请求直到有数据传递或超时才返回；客户端JavaScript响应处理函数会在处理完服务器返回的信息后，再次发出请求，重新建立连接；当客户端处理接收的数据，重新建立连接时，服务器端可能有新的数据到达；这些信息会被服务器端保存直到客户端重新建立连接，客户端会一次把当前服务器端所有的信息取回。

iframe是很早就存在的一种HTML标记，通过在HTML页面里嵌入一个隐藏帧，然后将这个隐藏帧的SRC属性设为对一个长连接的请求，服务器端就能源源不断地往客户端输入数据。

3）Widget

Widget是互联网应用的产物，是Web 2.0时代衍生的产物，真正地满足了消费者方便、快捷获取信息的需求，改善了人们使用互联网的体验，促进了互联网与移动通信的融合发展。它更加开放，更便于信息获取，在互联网中的作用主要体现在以下两点：首先，实现了

桌面应用和网络服务的结合，用户可以不用从浏览器登录站点就可以获取网络信息；其次，它提供了一个平台，用户可以自由地创建、发布、共享各类业务应用。

Widget 的 W3C 定义如下："Widget 是这样一个终端用户的概念：为了一些单一互动的应用，如显示或者更新本地数据或者网络的数据，而被封装成独立的，可以被下载和安装到用户计算机或者移动设备上的文件包。一个 Widget 可以是独立运行的应用(独立于浏览器之外)，也可被嵌入到 Web 文档中使用。"从功能上说，Widget 是一组工具集，提供包括时钟、日历、便签、天气预报、股票行情等各种小工具，给用户提供便利以及美化桌面的功能。

一般意义上的 Widget 包括页面 Widget、桌面引擎 Widget，通常所说的移动 Widget 可以认为是一种基于移动终端的桌面引擎 Widget。页面 Widget 基于浏览器技术，运行于页面上，用户游览页面的时候运行，用户可以在自己的个人页面上任意位置添加各种功能的 Widget，丰富页面的表现，例如用户可以把新闻、游戏、娱乐等信息整合到自己的 Web 应用上，也可以用来装饰 Web 应用；用户还可以把从某个 Web 应用上搜集的数据置入另一个页面中，比如把定制化搜索框加入到用户的博客中，或把某个 YouTube 视频加入到某位用户的 MySpace 页面之中，甚至创建一个综合了用户 Gmail、RSS 供应及网络相册等内容的完整 Widget 页面。桌面引擎 Widget 基于终端引擎技术，运行于个人计算机或手机终端上，用户运行后始终呈现在用户桌面上，其优点是它能同一时间接收来自不同信息源的信息。桌面引擎 Widget 的应用使得软件服务商可以推送各类资源给用户，本地或远程的软件服务可以十分方便地更新 Widget 内容，从而主动地将信息推送到用户的桌面，而不需要用户去启动软件本身查询，这就给广告商和运营商带来极大的商机。

在电信运营商中，KDDI、NTT DoCoMo 推出内置移动 Widget 业务的定制终端；在终端厂家中，Nokia、Apple 开发出各自的移动 Widget 平台；在业务提供商中，如 Google、YouTube 则关注 Widget 的应用。对于运营商而言，可利用 Widget 技术将已有的电信业务以一种全新的模式呈现给用户，并由此带来全新的体验。

随着 Widget 应用的不断演进，国际上很多 Widget 产品已经开始在手机媒体上应用。Nokia 推出了 S60 平台的 Widget，苹果的 iPhone 也搭载了 Widget，Google 的 Android 也提供了很多好用的 Widget。然而由于规范的不统一，各个厂家的 Widget 应用还不能做到互通。

13.2.2 新的计算模式

随着互联网的发展，支持大型计算和服务的模式不断发展变化，如网格计算和云计算，利用网络上各种资源，完成各种任务。

1. 网格计算

网格是一种信息社会的网络基础设施，是利用互联网把分散在不同地理位置上的多个资源，包括计算资源、存储资源、通信资源、软件资源、信息资源、知识资源等，全面连通和统一分配、管理及协调起来，通过逻辑关系组成一台"虚拟的超级计算机"。网格计算通过利用大量异构计算机(通常为桌面)的未用资源(CPU 周期和磁盘存储)，将其作为嵌入在分布式电信基础设施中的一个虚拟的计算机集群，为解决大规模的计算问题提供了一个模型。网格计算的本质是在一个动态的、多个机构的 VO(Virtual Organization，虚拟组织)间的资源

共享和协同问题求解，其焦点是支持跨管理域计算的能力，这使它与传统的计算机集群或传统的分布式计算相区别。现有的应用网格系统如美国科学网格（DOE Science Grid）、远程分布式计算与通信（Distance and Distributed Computing and Communication，DisCom2）、地球系统网格（Earth System Grid II，ESG）、TeraGrid、国家地震工程仿真格网（Network for Earthquake Engineering Simulation Grid，NEES Grid）、CrossGrid、天体物理虚拟天文台（Astronomical Virtual Observatory，AVO）、英国国家网格（U. K. National Grid）、德国的计算资源统一接口项目（Uniform Interface to Computing Resources，UNICORE）、亚太地区网格（APGrid）。

网格计算具有以下与一般网络不同的特点：大量动态的用户群体，大量动态的资源，计算能力动态可伸缩，多种通信机制，不同的本地安全解决方案，不同的本地信任机制，跨组织和区域的用户和资源。即网格计算具有大规模、开放、分布、异构、动态和可扩展等特性。对于最终用户或应用程序来说，网格看起来就像是一个巨大的虚拟计算系统。

从功能上来说，可以将网格分类为计算网格和数据网格。计算网格指的是一个广域范围内的一体化的集成和协同计算环境，其目标是利用网络中现有的软硬件资源，实现高性能计算的有效聚合，支持广域分布的高性能协同计算，解决大规模的科学计算问题。数据网格指的是将超级计算机的数值计算和分析能力与海量数据管理技术有机结合起来，从而为科学应用在分布式异构计算环境中实施资源发现和信息发现提供支持。具体而言，数据网格通过提供一组服务来支持资源和信息发现，通过存储资源代理使计算可以在异构的存储资源上进行。

网格计算技术的发展从 Globus Toolkit 的许多科学应用的部署，到 2005 年的 OGSA（开放式网格服务体系）标准的出现和相关应用的部署，再到可管理的（Managed）和共享的虚拟系统，其应用越来越广，而且也越来越被人们所接受。OGSA 的设计原则是从面向服务到虚拟资源；采用标准的接口服务机制，多协议绑定，本地和远程是透明的；网格服务机制，具有可靠和安全的模型，提供生命周期管理和发现等服务；多种宿主机环境，如 Java EE、.NET 或是 C。

网格计算和 Web 服务相结合面临着巨大挑战，网格计算的众多规范使得其和 Web 服务工具难以集成。新的框架 WSRF（Web 服务资源框架）直接基于 Web 服务，并强调语义解析，实现语义互操作。

2. 云计算

云计算是一种动态易扩展的资源计算方式，是网格计算、分布式计算、并行计算、效用计算、网络存储、虚拟化、负载均衡等计算机技术和 Internet 技术发展融合的产物，旨在通过 Internet 把多个成本相对较低的计算实体整合成一个具有强大计算能力的完美系统，即以 Internet 为中心的一个大系统，使计算分布在大量的分布式计算实体上，并借助基础设施即服务（IaaS）、平台即服务（PaaS）和软件即服务（SaaS）等先进的商业模式把这强大的计算能力分配到终端用户手中。也就是说，云计算是一种基于 Web 的服务，用户只需在需要时通过 Internet 访问服务，并只为自己所需要的服务付费，无须了解云内部的细节，不必具有云内部的专业知识，也不必直接控制基础设施。

云计算的主要特点如下。

(1) 超大规模。超大规模的云赋予用户前所未有的计算能力,如 Google 云计算已经拥有 100 多万台服务器,Amazon、IBM、微软、Yahoo 等的云均拥有几十万台服务器。企业私有云一般也拥有成百上千台服务器。

(2) 虚拟化。云计算支持用户在任意位置使用各种终端获取应用服务。用户无须了解所请求的资源和应用运行的具体位置,只需要一台笔记本或者一部手机这样的设备,就可以通过云提供的服务来实现自己需要的一切任务。

(3) 高可靠性、可用性和可扩展性。云计算系统使用数据副本容错、计算结点同构互换等措施保证服务的高可靠性。云计算也不是只针对特定应用,可以构造出各种各样的应用并支持其运行。云计算规模可以动态伸缩,满足应用和用户规模变化的需要。

(4) 自治性。云计算系统是一个自治系统,系统的管理对用户来讲是透明的,不同的管理任务自动完成,系统的硬件、软件、存储能够自动进行配置,从而实现对用户按需提供。

(5) 价格低。云的特殊容错措施、自动化集中式管理和租用模式等,使大量企业无须负担日益高昂的数据中心管理成本,资源利用率大幅提升,因此较低的成本就能快速高效地完成较大的任务。

很多 Web 应用因云而存在。同时,开发者可以利用云计算的能力和影响,进行 Web 应用的开发。开发人员以集中式 Web 应用的形式开发云服务,降低部署和维护成本。如《纽约时报》使用 Amazon EC2 实例在 36 小时内处理 TB 级的文档数据,而如果没有 EC2,这样的处理将要花费数天或者数月的时间。

目前,Amazon、Google、IBM、Microsoft、Sun 等公司提出的云计算基础设施或云计算平台,虽然比较商业化,但对于研究云计算确实比较有参考价值。主要的云计算平台有 Google 的云计算基础设施、IBM"蓝云"计算平台、Sun 的云基础设施、Microsoft 的 Azure 云平台、Amazon 的弹性计算云以及八百客公司的 800APP 等。

云计算并非网格计算。网格计算中,应用和(或)文档存放在组织的一个服务器上,通过组织的网络进行访问。云计算比网格计算的内容更广,涉及多个公司、多服务器和多网络,而且云服务和存储可以从世界上任何地方通过 Internet 连接进行访问。尽管云计算还处于起步阶段,但是其巨大的发展潜力与在 IT 界被强力追捧展示了其巨大的生命力。

13.2.3 多渠道访问

移动互联网的发展,使用户可以通过手机、PDA、便携式计算机、专用移动互联网,或者其他手执或车载等设备作为终端通过移动通信网络(2G、3G、E3G 等)或无线局域网作为接入手段,或直接通过 WAP 协议访问互联网并随时随地使用互联网业务。目前除文本浏览、图铃下载等基本应用外,移动互联网所提供的音乐、移动 TV、视频、游戏、即时通信、位置服务、移动广告等应用增长迅速,并仍在继续衍生出移动通信与互联网业务深度融合的其他应用。

与固定有线互联网相比,移动互联网和固定有线互联网的主要区别在于:终端和接入网络以及由于终端和移动通信网络的特性所带来的独特应用。具体而言,就是它的随时随地和充分个性化。移动互联网业务的特点不仅体现在移动性上,可以"随时、随地、随心"地享受互联网业务带来的便捷,还表现在更丰富的业务种类、个性化的服务和更高服务质量的保证。当然,移动互联网在网络和终端方面也受到了一定的限制。总的来说,其特点主要表

现为以下几个方面。

(1) 移动性。如手机等移动设备具有随时随地网络连接以及精确的位置信息，移动用户可实现无处不在的通信能力。

(2) 个性化。表现为终端、网络和内容和(或)应用的个性化。终端个性化：表现在消费移动终端与个人绑定，个性化呈现能力非常强，移动互联网终端一般是智能手机、PDA等，较普通个人计算机而言更具有个性化特点。网络个性化：表现在移动网络对用户需求、行为信息的精确反映和提取能力，并可与Mashup等互联网应用技术、电子地图等相结合。互联网内容和(或)应用个性化：表现在采用社会化网络服务(SNS)、博客、聚合内容(RSS)、Widget等Web 2.0技术与终端个性化和网络个性化相结合。

(3) 终端和网络的局限性。由于移动互联网采用无线接入，频率资源有限，接入速率普遍低于传统有线接入方式。而终端，不同于传统的笔记本及其采用3G上网卡的模式，受到终端大小、处理能力、电池容量等的限制。

(4) 业务与终端、网络的强关联性。移动互联网业务内容和形式需要适合特定的网络技术规格和终端类型。

(5) 业务使用的私密性。移动互联网业务及其内容和服务的使用更加私密，如手机支付业务等。

移动互联网的局限性尽管在一定程度上影响了Web应用的可用性，但是随着3G网络的部署，以及移动终端技术的持续提升，加之便捷的特性，移动互联网将会迅速发展。移动互联网的快速发展离不开优势业务的强力驱动，手机支付与电子导航这两大应用自然成为人们关注的焦点。移动互联网仍然是Web应用的发展方向，发展前景会越来越好。移动Web应用会日益丰富和完善。

13.2.4 Web操作系统

Web操作系统(Web-based Operating System，WebOS)是旨在使用户通过浏览器，便可在任何接通网络的计算机上使用的软件平台。即WebOS基于Internet和Web浏览器，本地计算机只需要支持浏览器，而不受本地计算机软硬件资源的限制。整个WebOS在一个Web页面上运行，可以通过任何设备在任何地方访问在这个WebOS上运行的Web应用来实现本地桌面操作系统上的各种操作，如创建、编辑和存储文档，播放媒体，等等。

现今国内外有很多WebOS，如基于AJAX技术的YouOS，支持多种语言尤其是简体中文的eyeOS，基于Flash的Goowy，界面类似于Linux的Desktoptwo，国内的WebOS如仿Windows XP操作系统界面的WGOS，原创EPOKOS，基于第三方应用集成的TomOS，千脑，等等。大多WebOS都是基于云计算实现，如EyeOS、amoebaOS、myGoya和CorneliOS等。

WebOS给基于它的Web应用开发带来诸多变化。WebOS开放相应的API，应用开发人员基于这些API，使用各种Web开发技术，如HTML5、JavaScript及CSS等，开发自己的各种Web应用，如联系人列表、音乐、日历、照片阅读器等。同时，可以使用相关框架进行开发，如基于IBM定义的iWidget规范，可构建一个基于Web的桌面操作系统风格的应用。

WebOS唯一的缺点就是受到网络速度的影响。为了实现用户入网的数字化、宽带化，提高用户的上网速度，光纤到户是用户网今后发展的必然方向。只要网络速度足够快了，用户能够使用万兆网，甚至百万兆网，到时WebOS定会成为现实。

13.3　语义 Web

Tim Berners-Lee 在 2001 年给出定义：语义 Web(Semantic Web)是现有 Web 的扩展，其中信息被赋予了形式化定义(Well defined)的语义，从而使计算机能够更好地与计算机以及人之间协同工作。这一定义说明了以下几点。

(1) 语义 Web 是当前 Web 的扩展，而非一个新 Web。

(2) 信息被赋予了形式定义良好的语义，使用元数据描述这些语义。

(3) 相关工具在这些元数据上进行操作。

Tim Berners Lee 在 2006 年对语义 Web 的说明：语义 Web 不只是将数据放在 Web 上，而是建立语义关联，使人与机器能够发掘 Web 上的数据，有了相互 Linked Data，当你有其中一些数据时，就可以找到其他相关的数据。语义 Web 的核心思想是：信息提供者在 Web 页面中采用语义标记(Semantic Mark up)；开发智能软件代理(Intelligent Software Agent)以搜索和处理 Web 页面中的语义标记，如信息代理、搜索代理、信息过滤等；Web 内容和软件代理的创建者通过本体(Ontology)相互理解，并使机器理解内容。

W3C 将语义 Web 看做是 Linked Data 的 Web，语义 Web 技术能够使人们在 Web 上创建数据、构建词汇，并编写处理数据的规则。Linked Data 由如 RDF、SPARQL、OWL 和 SKOS 等技术来实现。

语义 Web 的目标是提供统一的框架，以解决在不同应用、企业和社区之间的数据共享和互操作性问题。这种不同环境、异构、动态、开放以及全球化 Web 上的互操作性通过语义来保证。语义 Web 中的计算机能利用自己的智能软件，在 Web 上的海量资源中找到用户所需要的相关信息。

13.3.1　语义 Web 架构

1. 语义 Web 层次模型

Tim Berners-Lee 在 XML 2000 大会上提出了语义 Web 分层架构，并逐步细化，在 2006 年更新后的语义 Web 层次模型如图 13.3 所示。

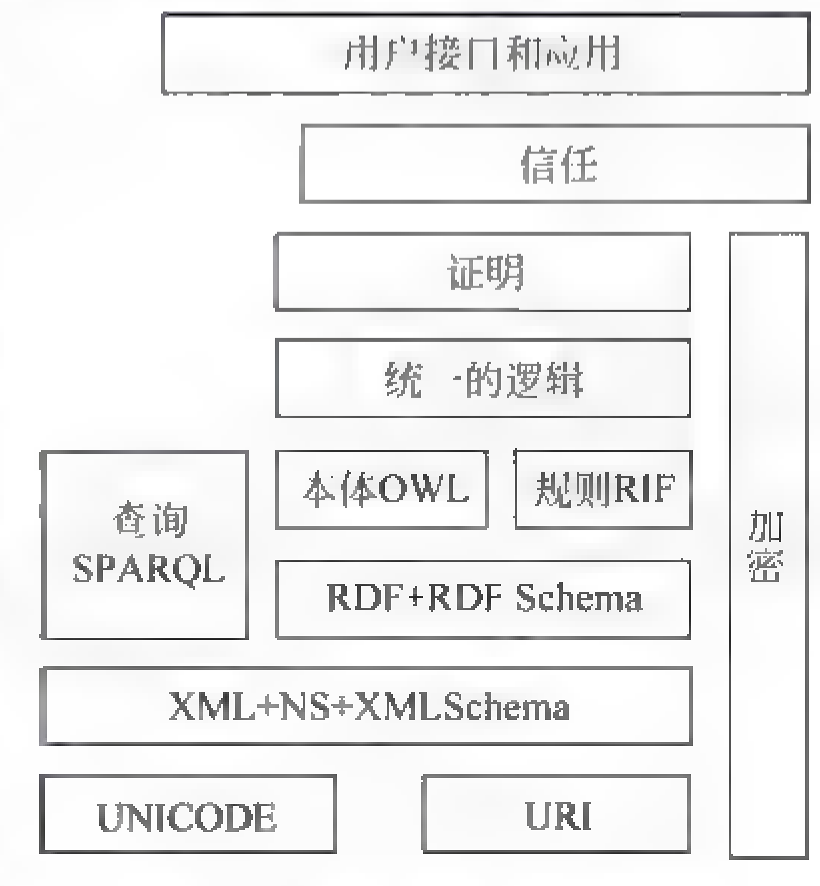


图 13.3　语义 Web 层次模型

(1) UNICODE 和 URI 层。UNICODE 和 URI 是基础，前者处理 Web 上资源的统一编码格式，保证使用的是国际通用字符集，有利于不同国家、不同民族的不同字符集在语义 Web 上的统一操作、存储和检索；而后者是统一资源标识，是统一定位符 URL 的超集，支持语义 Web 上对象和资源的标识，如 http：/www.w3.org/People/Berners-Lee 指的就是语义 Web 的创始人 Tim Berners-Lee 的信息。

(2) XML＋NS 层。该层包括 NS(NameSpace，命名空间)和 XML Schema。NS 由 URI 索引确定，目的是避免不同的应用使用同样的字符描述不同的

事物。XML是一种简单灵活的可扩展标记语言，特别适合于Web的文档进行交换。XML Schema表示共享术语和由人们制定的机器执行的规则，提供XML文档的结构、内容和语义的定义，提供一套完整的机制以约束XML文档中标签的使用，能更好地为有效的XML文档服务并提供数据校验机制。通过XML标记语言将Web资源的结构、内容和数据的表现形式进行分离，支持与其他基于XML的标准进行无缝集成。

(3) RDF+RDF Schema层。XML并不能表达机器可理解的形式化的语义，为此语义Web引入了RDF。RDF(Resource Description Framework)是语义Web的基本数据模型，用于描述Web资源，具有简单、开放、易扩展、易交换和易综合等特点，其目标是建立一种供多种元数据标准共存的框架。RDF数据模型不依赖于XML，但是遵守XML的语法。RDF Schema使用一种机器可以理解的体系来定义描述资源的词汇，是构造本体的初始语言，提供词汇嵌入的机制或框架，定义了将Web对象组织成层次结构的建模原语，主要包括类、属性、子类和子属性关系、定义域和值域约束。

(4) 本体层。本体是"一个对于共识的、已经概念化的事物的规范而明确的定义"，用于描述各种资源之间复杂和丰富的语义信息，即把现实世界中的某个领域抽象成一组概念(如对象、属性、进程等，如大学、职工、学生、学科和课程等)及概念之间存在的关系(如教师是职工的子类，X teaches Y等)。本体所描述的信息具有了计算机可理解的语义特点。基于本体的信息交换是实现本体之间的映射关系。因此，基于本体可以进行语义层次上的互操作，实现知识的共享和重用。在语义Web体系结构中，本体的作用主要表现在概念描述、语义揭示、一致性和推理支持。

(5) 逻辑层。提供智能推理的规则，为智能推理提供基础，可以增强本体语言的表达能力，并允许创建特定领域和应用的描述性知识。逻辑层是代理对用户任务进行分解、定位、协调、验证乃至最后建立信任关系的基础。

(6) 证明层。为保证代理工作的可靠性而提供的一种验证机制，支持代理间通信的证据交换，涉及实际的演绎过程以及利用Web语言表达证据，对证据进行验证等。证明注重认证机制，执行逻辑层的规则，并结合信任层的应用机制来评判是否能够信任给定的证明。

(7) 信任层。提供信任机制，保证用户Agent在Web上提供个性化服务，以及相互之间的交互安全可靠。基于可信Agent和其他认证机构，通过使用数字签名和其他知识才能构建信任层。Agent操作的安全性与用户对其的信任是语义Web的重要基础。

2. 语义Web基础

语义Web体系结构各层的内容共同构成了语义Web的技术、知识和逻辑三方面的基础。

(1) 技术基础。体系结构中的每一层都包含了为实现语义Web所需的技术(如图13.3中所示)，完成不同层的功能，各层逐级扩展，相互融合、补充。其中关键的技术有XML、RDF、Ontology，以及加密。

(2) 知识基础。语义Web描述事实性、术语和推理三个层次的知识，分别描述客观现实、概念及其相互之间关系的语义、推理规则。

(3) 逻辑基础。通过形式化的方法，借助强有力的形式化工具显式地揭示和描述语义Web中的语义，如RDF和OWL。

在语义 Web 的三个基础层面中，技术基础是该体系结构所直接体现的基础层面，知识基础和逻辑基础则是隐藏在该体系结构中间接体现的、深层次的基础层面。

3. 语义 Web 的优点

语义 Web 是数据的 Web(Web of Data)，其中，数据是 W3C 的链接的数据(Linked Data)的 Web 的构想。具有日期、标题、化学属性等数据，以及人们可能想到的任何数据。RDF 提供了将数据进行发布和链接的基础。如果将语义 Web 看做是一个全球数据库，那么，就需要相应的查询语言查询数据，SPARQL 是语义 Web 上的查询语言。

语义 Web 最大的优点是计算机对网络信息的"理解和处理"能力。"理解"是语义 Web 工作的第一步，因此数据意义的组织尤为重要。在"理解"之后，根据已有的数据和规则进行逻辑推理和自动处理。例如对于一个用于电子商务的购物代理，当用户把购物需求提交给代理程序以后，它会自动搜索符合用户条件的商品，并比较其中的不同，帮助决定目标商店，在验证目标商店的真实性与可靠性之后主动提交订单。

4. 开发工具

语义 Web 的各种相关研究工作和实际应用不断开展，出现了很多研究和开发工具。这些工具主要分为如下几类：RDF 编辑器，如 Altova SemanticWorks；RDF 解析器，如 rssparse；RDF 数据库接口工具；Topic Maps 创建及管理工具；本体构造工具，如 Ontolingua Server、OntoSaurus、WebOnto、Protégé-2000、WebODE、Oiled、OntoEdit、KAON、OntoEdit 和 DUET 等，提供一系列的工具以支持构建领域模型，以及具有本体的基于知识的应用；本体合并工具与集成工具，如 Chimaera、FCA-Merge、Protégé-PROMPT、ODEMerge 等；基于本体的注解工具，如 AeraDAML、COHSE、MnM、OntoMat-Annotizer 和 SHOE Knowledge Annotator 等；本体评价工具，如 OntoAnalyser、OntoGenerator 和 ONE-T 等；本体的查询与存储工具，如 ICS-FORTH RDFSuit、Sesame、Inkling、rdfDB、RDFStore、Extensible Open RDF(EOR)、Redland、Jena、TRIPLE 和 KAON Tool Suite 等；本体学习工具，如 OntoLearn、OntoBuilder、Text-To-Onto 和 Hasti 等。

13.3.2 语义 Web 应用

语义 Web 不断得到应用，应用系统或原型系统也不断出现，如 myGrid、MIAKT 和 SWAN 等项目。W3C 致力于采用语义 Web 技术改进各种不同行业之间协作、研究、开发和革新，如医疗保健、生命科学、电子政务和能源等，即垂直应用(Vertical Applications)。语义 Web 技术的应用有很多，其应用主要是采用本体对数据进行标注以利于进行语义推理(如采用 OWL、RDF 和 SKOS 等)，实现基于语义的检索与挖掘、领域建模、模式映射、语义标注、企业间的数据交换和知识管理、网络计算、分布式计算、基于语义的 SOA、电子商务、电子政务、语义门户以及语义社会网络等。

1. 语义 Web 搜索与挖掘

Web 上的信息容量越来越大，面对如此浩瀚的信息，传统的 Web 信息表示方法使信息检索的查准率和查全率均不理想，很难满足用户日益增长的需要。其根本原因在于当今

Internet的信息主要是基于语法构建的，而搜索引擎是基于关键字的，关键字的多义性和同义性降低了搜索的准确性，在搜索时通常会找到大量的与目标无关的内容。如新闻系统中搜索"NBA击败湖人"有关的新闻，可能返回大量有关湖人击败其他球队的页面，而这与搜索请求的本意相差太远。

语义Web技术可以较好地处理这些问题，通过进行语义标注的机制，即本体，根据精确的概念及其之间的关系、知识结构和推理规则进行搜索，从而得到与请求目标相近的结果。从Web上保留的大量普通页面中提取出语义信息，构建出页面内容本体，然后根据本体对新的页面进行语义标注。可采用本体自学习系统实现这一过程，达到本体的自动或半自动获取。文本信息可以采用语义Web技术，结合模式识别和对象提取等技术，实现基于概念的检索。如上例返回的结果是与"击败湖人"有关的新闻。现在已有的语义搜索引擎有Swoogle、TUCUXI、SHOE、OntoBroker、OntoSeek、WebKB、Corese等。

语义Web挖掘旨在将Web挖掘和语义Web这两大领域结合起来，使其相互促进，共同发展，提高信息获取的智能化程度。Web挖掘的结果有助于构建语义Web；语义Web的语义知识使Web挖掘更易实现，同时能改善Web挖掘的结果。在语义Web中超链接显式表达出来，这促使对Web结构进行更进一步的挖掘；同时，Web页面内容具有了明确的语义要求能够接受更加结构化的数据输入的Web挖掘技术。语义Web挖掘分为：语义Web内容及结构挖掘和语义Web使用挖掘。

2. 知识管理

知识管理(Knowledge Management，KM)关注组织中的知识的获取、访问和维护，是组织中大型企业中的一个关键活动。越来越多的企业将知识看做是智力资产，它可以提供生产率，创造价值，增加企业自身的竞争力。因此，知识管理成为企业的战略工具。

1）知识管理技术的不足

目前，大部分知识的结构性较弱，如文本、音频和视频等。从知识管理的角度而言，目前的技术具有很多弱点，表现如下。

① 信息搜索仍依赖基于关键字的搜索引擎。

② 信息抽取时需要人为参与来获取关联信息，智能代理还不能很好地满足需求。

③ 信息维护中仍然面临术语不一致和过期信息难以移除的问题。

④ 采用数据挖掘进行信息恢复对分布式、结构性差的文档仍然难以胜任。

⑤ 信息浏览的访问控制在内联网或者Web上难以实现。

2）语义知识管理的优点

语义Web技术，特别是借助于本体和机器可处理的元数据，对知识进行精确的语义标注之后，使知识管理系统具有如下优点。

① 知识通过其含义在概念空间中进行组织。

② 自动化维护工具支持一致性检查和新知识的抽取。

③ 以用户友好的方式回答查询将取代基于关键字的搜索。

④ 支持跨越多个文档的回答。

⑤ 支持文档或文档中的部分可访问性的定义，即谁可以访问哪部分信息。

3. 语义网格

在英国的 e Science 计划研究中，人们发现，网格的现有努力和 e Science 设想之间存在差距。要达到 e Science 的易用性和无缝自动化要求，必须实现尽量多的机器可处理性和尽量少的人类介入，这和语义 Web 的目标有一定的相似。在 2001 年最先提出了语义网格的构想，并于 2002 年在全球网格论坛 GGF 成立了语义网格研究组 SEM GRD。语义网格小组对语义网格的定义如下：语义网格就是"对当前网格的一个扩展，其中对信息和服务进行了很好的定义，可以更好地让计算机和人们协同工作"。

语义网格构想的关键是把所有的资源，包括服务，都用一种机器可处理的方式进行描述，旨在实现语义互操作性。为了达到这一目标，把语义 Web 的关键技术应用到网格计算中，下至基础设施，上至网格应用，使计算机尽可能取代人进行网格上的信息处理，从而让诸如电子商务、电子政务、数字图书馆等智能化服务在网格上开展。语义网格方面的研究重点解决 3 个方面的问题：资源的规范组织、语义互联和智能聚合。

4. 基于语义的 SOA

SOA 是一个组件模型，将应用程序的不同单元称为服务，通过定义良好的接口和协议联合起来。从 1995 年发展至 2010 年，SOA 技术从 CORBA/COM、JINI、SOAP、WSDL、OWL-S、WS-*、REST，到 2010 年云的概念和链接的开放数据(Linked Open Data)，即从 Web 服务描述发展到本体驱动(语义 Web)的 SOA。本体驱动的 SOA 旨在采用 RDF、OWL 和 OWL-S 等标准语义 Web 方法，实现 Web 服务的自动发现、调用和组合，使数据在不同数据源之间可进行语义互操作。其中，用户和程序通过语义业务层进行交互，语义 SOA 层定义标准词汇、形式化模型(企业本体模型和适配器本体模型)和数据源之间的语义关系，语义层提供数据的一致视图、查询和其他服务。

NASA NexIOM 和 CxDA 使用 OWL 本体策略支持 XML 和其他数据格式的互操作；NASA TCMX 定义了基于本体的控制词汇表，并构建了 SBFI 和 QUDT 两个系统的本体。数据质量对于 SOA 而言至关重要，因此需要有命名和标识规则，需要有词汇管理和转换的本体。

5. P2P

P2P 与语义 Web 技术的结合将会使目前基于本体的知识管理解决方案进一步去中心化。目前的 P2P 系统有的提供简单语义结构，如 Napster 和 Kazaa，把语义标注和集成的任务留给用户；有的系统甚至根本没有提供语义支持，如 Gnutella 和标准的基于 DHT 的系统。分布式信息系统的语义互操作能力是其可用性的一个首要关键因素，是结构化、分布搜索、数据交互和集成、高层服务和处理的重要前提。

P2P 系统架构中解决语义互操作的技术如 LAV 或 GAV 等，依赖于全局预定义的模式，这对于全球范围的 P2P 环境而言，并不现实。因此，期望并不依赖于中央协调和知识库的方式，通常称为结点数据管理系统(PDMS)，采用结对映射在异构的邻居之间进行映射。如果不同的邻居组中模式不同的结点要交换信息，可以相互进行模式映射。从而，可以进行基于语义的路由。

6. 电子商务

传统的电子商务存在控制、信息发现、环境、兼容性和智能性等许多问题。如通过包装器(Wrapper)实现在网上商店中的信息抽取功能时，每个商店需要一个包装器，开发代价高，而且目前大多采用基于关键字的抽取方式，如“价格(Price)”、“$(Y)”后接正数等。但是这种规则式无法保证很好的效果，如运输费用和时间、安全条款等也是影响用户做出决策的有用依据。

在电子商务应用中引入语义 Web 技术能在一定程度上解决上述问题。通过开发软件代理(Agent)对产品信息和服务条款等进行解释。如在 B2C 系统中，可做到以下几点。

(1) 价格和产品信息可以被正确抽取，运输和安全条款也可以根据用户需求进行解释。

(2) 获取如网上商店的评价等其他信息。

(3) 不再需要开发底层包装器。

(4) 购物代理更加先进，可以和商店代理进行自动协商，商店代理可以通过逻辑推理出什么样的客户拥有什么样的折扣率。

再如在 B2B 系统中，采用语义 Web 技术可以消除传统只有专家能懂的电子数据交换(EDI)方式的复杂性以及相应的实现和交互代价，举例如下。

(1) 易于企业之间的低代价交互。

(2) 采用抽象领域模型(Abstract Domain Model)解决术语差异。

(3) 采用转换服务进行数据交换。

(4) 使用软件代理进行自动(或半自动)的拍卖、协商和起草协议。

语义 Web 技术在电子商务中应用的研究主要集中在描述语言、基于本体的企业商务集成方法与架构(用户需求或产品或企业商务过程的语义描述、本体映射、Web 服务发现与组合、电子商务集成框架)、领域本体的管理(包括电子商务本体构建、本体库的管理)、CRM、电子交易等方面。

7. 语义门户

语义门户是语义 Web 技术驱动的门户 Web 应用，是实现具有共同兴趣目标的用户之间的信息交流和共享的平台。语义门户使用语义 Web 技术来提供语义检索、浏览和内容集成。与传统门户相比，使用语义 Web 标准进行门户设计具有如下几个方面的优点。

(1) 搜索途径。通过完备的领域本体实现多维检索和浏览。

(2) 信息组织。信息组织结构具有可扩展性，使用自下而上的设计和分散式更新。

(3) 用户操作范畴。领域用户可以添加新的类别和组织结构，扩展信息结构。

(4) 内容管理。分散式管理门户内容，同一资源隶属于不同聚合并且拥有多种解释。

(5) 实现机制。信息结构可被机器直接读写，有利于跨学科门户的资源整合。

(6) 信息的发布与维护。信息发布者可在多个门户之间使用通用表单来发布信息，资源以再利用方式提供给多个门户，信息的发布与维护由同一主体完成。

目前已有很多语义门户，如 OntoWeb、Esperonto、Empolis K42、Mondeca SWWS、ITM、Mindswap、OntoWebEdu 等。创建语义门户的方法和框架也有很多，比如 SEAL、ODESeW、K42、ITM、OntoWebber、Onto-Weaver 等。

8. 语义社会网

语义社会网旨在现有社会网之上(即 Web 2.0 之上)加入语义层，增强现有社会协作 Web 的指导功能和减少 Web 类似目录的作用。如 Flickr 通过采用语义 Web 技术的社会分类法(Folksonomy)，使其社会交互具有语义层次的支持。

13.4 Web 工程发展

Web 工程与关键工程、人机交互、人工智能等其他学科有很大的关联，并已经被看做是多学科领域。Web 工程随着万维网的发展而快速发展，越来越多的研究者、实践者、教育者等都加入到促进 Web 工程发展的行列。

技术会变化，工具会演化，更好的方法会出现，陈旧方法会逐渐消失，人却始终不变。如果我们想通过对过去半个世纪的观察来获得一些知识的话，那么这些知识即：大多数的技术问题都可以解决，但是与技术的发展和使用相关的人的问题却成为持续的挑战。

1. Web 工程发展面临的挑战

Web 应用的发展及其多样化和整合化，以及语义 Web 的发展，使 Web 工程向语义 Web 工程方面发展的过程中面临的如下挑战依然保持不变。

(1) 认识到利益相关者的需求是极为重要的，但与此同时要明白我们交付高质量且健壮的解决方案的快速程度是有限制的。

(2) 改进 Web 工程的过程，使其敏捷、可适应并且有效，过程不应该妨碍好的解决方案。

(3) 认识到一个期望的 Web 应用解决方案的特征经常会在开发过程中有所演化，并且认识到创建一个正确的解决方案经常会包含存在于新系统之间复杂的相互影响和现有处理及行为的变化——而且很多变化直到系统被应用以后才会完全明了。

(4) 开发方法和技术让我们对问题有更好的认识，对基于 Web 的解决方案有更好的理解。要认识到方法和技术不能替代人员沟通和对问题有意义的理解。

(5) 为了适应 21 世纪业务环境的动态特性而对变更进行管理，同时避免不能控制的变更造成的混乱。

2. Web 工程的发展方向

事实上，现代世界中的每件事以及每个人都通过某种方式连接到 Web 上，并在网络上相互链接。Web 工程就是向提供更好服务的方向发展，主要包括如下几个方面。

(1) 搜索方面。搜索 Web 应用模型库，基于结构化和语义特性的近似 GML(通用置标语言)抽取，高级搜索中更高级和更好的自动查询建议服务，设计工程化搜索计算应用的服务市场，等等。

(2) Web 服务方面。工程化可视 Web 应用的自动控制器，有效管理多层 Web 服务的突发工作负载，管理 Web 服务等级协议标准，采用聚合模式和基于层次匹配的 Web 服务发现，开放和可扩展的分布式感知应用的 Web 消息服务，REST 的作用逐步增强。

(3) 开发过程方面。模型驱动 Web 应用的多层测试,使用 WebSpec 捕获和促进 Web 需求,遗留 Web 应用再工程为 RIA 应用,多种设备认知环境中从逻辑描述到处理语音接口,质量、实际可用性和用户体验作为 Web 应用评估的核心驱动力,为脚本提供接口(如 Greasemonkey)。

(4) Web 2.0 方面。采用上下文感知交互方法处理用户本地上下文,微博的进一步开放、分布和语义化,基于 Web 进行安全的元模型化(Metamodeling)远程模型访问的协作环境,领域无关的自动 Web 表单填写,可扩展和混搭的面向位置的服务。

(5) Linked Data 方面。灵活的基于规则的互连、集成和丰富用户数据的方法,对 Linked Data 进行评级(如 DBpedia),通过自动加入解除引用的语义标注丰富 Web 页面。

(6) 性能和安全性方面。对支持智能卡的 Web 应用采用统一安全代理,为流 Web 内容生成有效的团队云。

另外,工业界在不同方面逐步发展进步,如采用 REST 开发 Web 服务平台,WebRatio BPM 逐步成熟,出现了可视化进行快速企业应用集成的工具,公众网正在逐渐部署,等等,甚至是针对不同外包服务的人之间的交互方面也在不断实践和发展。

从 Web 工程师的角度而言,所采用的技术既让人兴奋,又具有更大的动态性。Web 工程师要继续在这种环境中工作,并确保 Web 更好地提供服务。

13.5 总结与展望

从 Web 工程的概念出现至今,短短的十来年中,Web 工程各方面的研究和应用已经取得了很大的进展和成果。但是新的 Web 平台、技术和标准,新的 Web 应用种类,网格计算、云计算等新的计算模式,多渠道访问模式,Web 作为 Web 操作系统这样一个大的软件平台,各种 Web 应用支持语义特性以达到机器可理解并可进行自动化处理的目的,等等,这些方面的不断发展演化,会进一步改变人们使用 Web 的方式。加之人们对 Web 应用服务的质量需求,使得 Web 应用的复杂度、灵活性更高,对机器可理解和可自动处理的要求更高。

Web 工程需要应对技术和标准等各方面的不断发展和演化,适应工具的演化和方法不断更新换代。因此,Web 工程在需求的获取、新的方法和技术、开发过程、快速变更的管理与控制等各方面仍有待于进一步发展。

Web 架构在演化,语义 Web 的发展与应用、XML 技术、服务概念的强化、多渠道的访问、浏览器和验证工具,以及 Web 设计技术和应用等多个方面也在发展和演化。随着这些方面的发展和演化,Web 工程也必将产生更多更新的技术、方法和工具,以支持 Web 更好地为世界和人类服务。

参 考 文 献

[1] Gerti Kappel, Birgit Pröll, Siegfried Reich, et al. Web Engineering: The Discipline of Systematic Development of Web Applications. John Wiley & Sons, 2006.

[2] Roger S Pressman 著. Web Engineering: A Practitioners Approach. Web 工程：实践者的研究方法. 霍秋艳,等译. 北京：机械工业出版社, 2010.

[3] WooJong Suh. Web Engineering: Principles and Techniques. Idea Group Publishing, 2005.

[4] Rossi G, Pastor O. Web Engineering: Modeling and Implementing Web Applications. Springer, 2008.

[5] Paul S Wang, Sanda S Katila. Web 设计与编程导论. 邱仲潘译. 北京：高等教育出版社, 2009.

[6] Extreme Programming Agentle Introduction. http://www.extremeprogramming.org. 2011-01-03.

[7] Tim O'Reilly. What is Web 2.0. http://oreilly.com/web2/archive/what-is-web-20.html 2005, accessed 2010.11.02

[8] Michael K Jones. Software Configuration Management for the Web. Methods & Tools, 2008(8): 16-25.

[9] Penny Mcintire. Web 视觉设计. 叶永彬,等译. 北京：机械工业出版社, 2008.

[10] Michael Miller. 云计算. 姜进磊,等译. 北京：机械工业出版社, 2009.

[11] Matthew David. HTML5: Designing Rich Internet Applications. Elsevire, 2010.

[12] Steve Krug. Don't Make Me Think: A Common Sense Approach to Web Usability. 2nd ed. Pearson Education, 2006.